THE ELEMENTS OF HYBRID ELECTRICAL SYSTEM DIAGNOSIS

A Thesis

Presented for the Degree of

Doctor of Philosophy

in

Engineering

in the

University of Canterbury,

Christchurch, New Zealand

by

Howard J. Jelinek

1969

ACKNOWLEDGMENTS

The material presented here has been developed over the four and one half years that I have been lecturing in Electrical Engineering in the University of Canterbury. I am indebted to many friends and my associates in the Department of Electrical Engineering for intellectual and moral support and for advice and encouragement. In particular, I am grateful to:

Professor L. Kay for encouraging research by example,

Professor J.H. Andreae for his enthusiasm, his astute and questioning analysis of some of my very rough ideas and for his willingness at any time to read and comment tersely on drafts of this thesis,

R.H.T. Bates for reading various chapters, and for lending encouragement and advice,

My parents and personal good friends who have suffered my ups and downs and made life enjoyable by unfailing support and understanding,

Lee Dawson and Sandra Dobson whose humour, patience and expertise in typing difficult material were indispensible,

Judy, my wife, for spending her honeymoon with a thesis.

INTRODUCTORY REMARKS TO THE READER

The subject of this thesis is the formulation of diagnostic techniques for detecting and isolating faults in hybrid electrical systems. The techniques are developed using model-variation studies.

Diagnostic techniques are valuable because they permit the engineer to *a priori* assess the performance of a system containing a malfunction. Furthermore, by knowing *a priori* how a fault will affect output signals, the engineer can design programmed equipment which will automatically monitor the system signals and detect the occurrence of a malfunction.

My primary overall objective in this investigation has been to formalize the approach used to develop diagnostic information for hybrid electrical systems. The approach is formulated in Chapter 1. The need for efficient diagnostic analysis methods has grown as systems have become more complex. The heuristic methods for system testing which have been applied in the past are no longer satisfactory. Diagnostic analysis covers the areas of test information generation and test procedure specification. These are complementary. Unless test information can be computed prior to the occurrence of actual system faults, there is no way of knowing for certain that the test procedures specified are effective.

Applications of diagnostic analysis are to test equipment specification and diagnostic data development. The latter application is essential to all diagnostic studies and is particularly useful for compiling fault dictionaries or maintenance charts which list the likely faults, given a set of symptoms.

The approach formulated in Chapter 1 is explicated in Chapters 2,3, and 4. Techniques are developed which are useful for deriving fault detection and isolation information and for specifying efficient tests. Methods for applying diagnostic models to analogue, combinational and sequential network diagnosis are developed in these chapters.

During the draft stages of this thesis, each chapter was written to "stand alone". Unfortunately repetition has not been completely eliminated from this draft and some unnecessary overlap occurs in various places. In addition, the presentation is largely informal in the sense that it is devoid of mathematical development. Because premature mathematical formalism can sometimes obscure the simplicity of the ideas, I have used words where symbols might well have abbreviated the discussion. Owing to the original drafting method and the style of presentation, the number of pages is rather more than might be expected or desired.

To the reader, I recommend that Chapter 0 be read quickly. Chapter 1 is long but the ideas are definitive and much of the material forms the bakcground for subsequent chapters. In Chapter 2, the first part may be skimmed. It contains some of the material in Chapter 1 masticated for the analogue network digestive system. The first thirty pages can be scanned. In Chapter 3, Sections 3.2 and 3.6 - 3.7 are largely original. Chapter 4 is short and can be read rather more quickly than the previous chapters.

CONTENTS

PROLOGUE

Many hundreds of thousands of people gain their livelihood from repairing mechanical and electrical devices rangeing from simple home appliances to complex data processing systems. These people, from the automobile mechanic and home appliance repairman through to jet engine and computing system maintenance engineers have one factor in common; they are all concerned with identifying the source of any malfunction and subsequently performing the necessary repair. Repair is accomplished by adjustment, modification or replacement of a part or parts.

Repair may be required directly following the first indication of disrepair or when the state of disrepair reaches a certain level. Disrepair can take many forms. For example, there can be catastrophic states of disrepair such as exhibited by a toaster whose element has open circuited or a degraded state as when the toaster fails to regulate satisfactorily. People tend to try to live for a time with the latter case while the former demands immediate action. In general, repair is initiated subsequent to the indication that one or more states of disrepair exist. We will, from here on, refer to disrepair states as faults, malfunctions, or errors. It is assumed that the error free condition (good design) is recognizable. The indication of a fault will be termed a symptom. A symptom may be defined as a qualitative or quantitative change in the observable functioning of the device, machine, or system. Observations may be visual, aural, kinesthetic or olfactory and they may be qualitative, or quantitative or a combination. The repair man may use one or all of these senses to make observations and from these, deduce

the likely cause(s) of the symptoms. Through a procedure which is equivalent to a sequential decision process, the physical changes which brought about the state of disrepair can be isolated; the fault can be located. Observing (or perhaps recognizing) a symptom which is indicative of a malfunction, will be termed fault detection. Using information from the observable symptom (effect) to locate the more basic (cause) of the symptom or perhaps another symptom, will be given the name fault isolation. This general class of activity will be called a diagnostic process or procedure. It may deal with fault detection or fault isolation or both. In this thesis, the diagnostic activity is regarded as encompassing all aspects which pertain both to fault detection and isolation. The words diagnosis, check-out, testing and fault studies refer to the general problem of malfunction identification.

This investigation is confined to the problems of formulating an approach and developing procedures for fault detection and isolation in hybrid electrical systems. Discussion is restricted to the sub-class of hybrid systems that can be realized by interconnecting analogue, combinational digital, and sequential digital networks. We further restrict analogue networks to be those whose component parameters are time invariant. This eliminates time varying parameter networks from the discussion. For the analogue network procedures which are described in Chapter 2, stationarity over the period during which the diagnostic procedure is being applied is assumed. In the first restriction, we do not eliminate many practical systems from the discussion; the latter assumption of stationarity is crucial to most procedures for system state estimation or parameter identification[148]. It is essential to realize that we

are not regarding physical parameters as eternally time invariant: If they were, things would last forever. Stationarity of parameter values over the period during which diagnosis is performed is all that is required for a parameter to be classified as time invariant. The word constant will sometimes be used to mean time invariant.

The need for improved system diagnostic procedures arises in almost every type of large system. Hybrid systems of the type that will be discussed here are being built which operate in hostile environments or under conditions where a continuous indication of their status is mandatory. Any malfunction must be located and repaired quickly. In these systems, tests must be made on-line using actual system operating signals or by introducing a very limited number of external stimuli. It is mandatory that fault detection circuitry and testing equipment be an integral part of the system. Most electronic equipment contain various indicator lamps which can be regarded as basic internal fault detection devices. It has been shown[116, 12] that external test equipment tends to induce faults in a system and causes malfunctions which would not have occurred in the untested system.

The importance of having effective diagnostic procedures has been recognized by the computing system manufacturers from the very beginning. Even with the vastly improved reliability of components, diagnostic engineering is one of the major areas of computing system design[31]. The design of hybrid systems is generally more complex than computing system design; hybrid systems may include computing systems as sub-systems. While a good portion of the computing system design has been automated, hybrid system design is empirical in many of its facets. This study is an attempt to

formalize one small aspect of the hybrid system design.

The diagnostic approach (methodology, formalism, philosophy) can be described as consisting largely of heuristic methods. These methods may be broadly described by the term check-out[147]. One of the first relevant references to appear in the literature which stresses the value of a more formalized approach is a paper by Brule, Johnson and Kletsky[19]. The more recent references have tended to relate the formalism to specific techniques for particular types of networks, see for example, Roth [123], Chang[23], and Kime[71]. Surprisingly, work on the general systems approach seems to have temporarily subsided. In Chapter 1, a formalism is developed which has the objective of defining a structure within which system diagnostic procedures can be developed.

The approach we formulate is basically a model-variation analysis. It is sufficiently flexible to be applied at the system, sub-system or component level. Model-variation analysis refers to a philosophy that regards the faulty sub-system or component as a modification of the good version. The model accommodates this modification by suitable specification and subsequent adjustment of a set of coefficients which we term parameters. These models are designed to be incorporated into simulations using general purpose digital computer programs. Using program methods, procedures for fault detection and isolation can be developed.

In this thesis, we imagine the total problem of developing diagnostic procedures as being divided into two parts. The first part will be called analytical procedures and the second is termed hardware procedures. In reality, no clear dichotomy exists. However, this division stresses a point of the philosophy developed

herein. We regard the problem of procedure definition as one of first defining how to do it and then of actually doing it. The "how" can for the most part be developed prior to or during the system design specification stage. The "doing" requires both knowledge from the first part and special test equipment to measure and interpret the information obtained during the second part. The first part which is regarded as an analytical and specification stage, will be called the a priori stage, indicating that it is done prior to the occurrence of any actual system fault. The second stage will be called the a posteriori stage. Information on test signals developed during the a priori stage can be applied in conjunction with custom designed hardware to achieve efficient and thorough fault detection and isolation on the real system.

Analytical and procedural methods which are developed in this thesis complement test equipment by specifying test types (e.g. input signal sequences), by giving pre-test interpretation to likely results of test measurements, and by developing measures for predicting the degree to which faults can be detected and isolated.

In the last five to ten years, references have appeared in the literature discussing particular analytical techniques for specifying tests on specific types of networks [7,34,40,81]. However, there are very few references on a priori analytical methods applicable to hybrid systems. The more common approach is to use heuristic or empirical methods. Most empirical methods are based on advice from the design engineer who uses intuition and experience to predict possible failure modes. His assistance and any other relevant information on failures largely determine design constraints on test equipment. For digital systems, check-out equipment will

apply such actions as bit examination, comparisons of registers, masking, and exhaustive instruction testing. (66) Power supply load regulation and amplifier bandwidth are two important analogue network characteristics often monitored by test equipment. In systems containing for example, servo equipment, special functions can be generated digitally by the test equipment computer if the frequency requirements are compatible with digital function generators. Normally, test equipment, both automatic and manual, is composed of stimulus generators, response recorders, magnetic tape and disc storage units and a CPU of some form along with the necessary control units.

Ideally, the a priori stage will: a) include the specification of tests to be applied by the automatic check-out equipment, b) state the necessary test sequencing procedures, and c) give a quantitative estimate of how thoroughly the system can be tested by the particular test procedures.

The thesis propounded herein is directed primarily to the a priori diagnosis problems. Four fundamental assertions which underlie the overall thematic development of this work are the following:

1. We postulate that any physical network can be regarded as a device which processes the inputs to obtain the outputs. The processing activity performed by the device can be described by a model or abstractly in terms of a mapping or a series of mappings provided the inputs and outputs are interpreted as sets or sequences. The mapping(s) depend on network structure, on physical laws relating structure and parameter values, on internal network signals, and on time.

2. Networks are classified in this thesis according to the type of signal information - continuous or discrete amplitude - they are designed to process. Analogue networks process

continuous signals and combinational and sequential networks process discrete signals. Intermediary networks which form interface functions (e.g. comparators, ADC's and DAC's) in hybrid networks can be treated using models developed in Chapter 1.

3. Mathematical models for electrical network components are developed which are useful for simulation of variation in the physical component characteristics. These models will be termed isomorphs. Networks can be modelled using component models and a one-to-one correspondence between model and physical parameters can be maintained. Alternatively, networks and sub-systems can be modelled and a set of parameters developed which mirror the variational characteristics of the more detailed model. This more coarse model will be termed a homomorph. In both types of models, parameter variations (perturbations, changes) are equated with certain types of faults. In Chapters 2,3 and 4 algorithms are developed for fault detection and isolation based on input-output-parameter variation information. In abstract model terminology, a fault condition is equivalent to a mapping error.

4. Fundamental to this thesis is the requirement that any proposed diagnostic procedure possess a computable measure of effectiveness. In other words, we regard a technique as useful only if we can determine this usefulness quantitatively. To say - this is an optimum method - has no significance unless its figure of merit has been computed and verified as being the best possible. The figure of merit used in this study is called testability (140 147) and is defined to cover both fault detection and isolation.

Improvement in next generation checkout procedures can be realized. Using the a priori techniques proposed in the following chapters, it should be possible to a) reduce testing time by utilizing methods which are non-exhaustive, b) obtain a quantitative measure of the amount of information accumulated on the system status for a given test and c) provide a consistent framework

for interpreting the a priori diagnostic information and for relating this information to the requirements of the a posteriori problem of developing hardware and software for automatic checkout.

In Chapter 0, the importance of diagnosis to the systems engineering approach is stated. In particular the influence of diagnosis on system effectiveness is illustrated. Effectiveness can be improved through diagnosis. The diagnostic methods introduced here are based on models which are amenable to parameter variation studies. In this chapter, the association between faults or malfunctions and model parameter value settings are illustrated. The types of faults and how they manifest themselves electrically are mentioned. A modelling process which is a necessary prerequisite for all a priori diagnostic studies is discussed.

Chapter 1 introduces some key historical developments in diagnosis and proceeds from this introduction by outlining a "diagnostic framework". Basic mathematical models that represent faults as changes in parameter value settings are described. Models of this type are used for developing test data and for studying various testing methods. We have devoted a substantial section of this chapter to the discussion of how the diagnostic models can be applied to developing test data and test procedures.

In Section 1.8., an example of the application of model-simulation, applied to a hybrid interface network is presented which illustrates the type of procedure that can be used to detect and isolate faults in an interface network.

Chapter 2 discusses the analogue network diagnosis

problem. Basic network models are first presented, then some fundamental limitations on using analytical techniques to compute parameter values from input output information are developed. The relationship between diagnosis, observability and identification is briefly mentioned. To apply the general testing concepts established in Chapter 1 to the time domain models, sampling of the network signals is required. Two methods for classifying sampledanalogue signals are given and interpretations of these signals and how they relate to network parameter value changes is established. This leads directly to discussions on the use of sampled signals in fault detection and isolation techniques. The chapter concludes with some examples of applications and a comparison of several frequency domain techniques.

Chapter 3 discusses combinational network diagnosis. Test efficiency measures which are very important for combinational networks are discussed and optimum tests for single and multiple output combinational functional elements and networks are described. Methods for displaying test information are mentioned and a new diagnostic model is introduced called the faultable gate. Applications of this model to test information development are described and example print-outs from a simulation program incorporating the faultable gate are shown for a multiple output network.

Chapter 4 discusses a technique for sequential network diagnosis. Relationships between abstract sequential machines and their physical counterparts, sequential networks, are explained. The differences between abstract machine experiments and physical network diagnosis are contrasted. Several new definitions are given and methods for developing fault

detection tests are presented. An empirical method equivalent to pseudo correlation discussed in Chapter 2 but applied to sequential networks concludes the chapter.

CHAPTER 0

0 SYSTEMS, FAULTS & MODELS

0.0 INTRODUCTION

The major aim of this chapter is to justify the systems approach to diagnosis. In particular we want to stress the advantages of employing diagnostic models for developing fault detection and isolation information. It will turn out that models are of two basic types. One is a mathematical or simulation model which is one-to-one with the network it represents. The other is termed a functional model. It is not necessarily one-to-one with the network but it must possess the potential to accurately represent input-output behaviour of the network or device both for the fault free condition and for specified faults. This model is representative of the class called simulation models.

In this chapter, we introduce the relationships between the general systems engineering problem and the problem of diagnosis in a class of systems. Those systems which are the object of this study have been loosely termed hybrid systems[28,33,136]. Hybrid systems are sometimes defined as those composed of electrical analogue and digital electrical equipment, mechanical, hydraulic, fluidic, electro-chemical and electro-mechanical equipment and the human being. We shall use the word equipment to describe hardware components (subsystems) of the system. An example of the general hybrid system would be the modern jet aircraft. In this investigation, discussion will be limited to those hybrid systems containing electrical analogue and digital equipment only. Although methods developed in later chapters can certainly be extended to develop fault detection and isolation methods

for a larger class of systems, including notably those containing mechanical and fluidic[130] equipment, we have confined this discussion to techniques which apply to systems made up from hardware composed of electrical analogue, and electrical combinational and sequential digital networks.

0.0.1 Background

Interest in how best to design, analyse, control, and maintain large systems has grown considerably within the last decade[33,48,85,117,145,148]. A direct attack on the problem was not seriously attempted until the period now known as the second world war. People working on logistics problems were the first to consider the operation and interactions of the total integrated transportation and supply systems[30]. The field of operations research has developed from this early effort. Many of the techniques in control theory and reliability which are the substance of contemporary research originated and grew from these early studies. General systems theory sprouted from this bed of knowledge.

Diagnosis can be identified as an area of the systems maintenance problem[63,48]. Most of the early work on system diagnosis is contained in unpublished company reports which appeared between 1952 and 1959. A list of these appears in the book by Goldman and Slattery on page 277[48]. Since 1959 the number of references in the general area of test procedures, testing techniques and testing equipment has increased rapidly. Testing techniques were discussed in a special issue of the IRE Transactions on Milatay Electronics in 1962[147] and more recently Bruer[18] has compiled a fairly comprehensive set of references on digital network diagnosis. Most of the diagnostic techniques are oriented

to solving a particular aspect of the diagnosis problem rather than looking at general objectives.

The objective of some recent systems research has been to develop models and analytical techniques for predicting the performance of large "super-systems"[9,85]. An objection to many of the models which have been proposed so far is that they are too general or too specific; often the general models bear little resemblance to reality, tending to obscure rather than clarify the physical relationships contained in real systems[94]. The specific models are one-off, relevant only to the particular system for which they were developed.

This criticism should not be construed as a denial of the worth of the system theorists work. Rather it is intended to reflect the difficulty inherent in developing total system models which represent all facets of the system structure and function.

An alternative to the total system model is a number of specific models, aimed at modelling one or more aspects of the system engineering problem. Text books have been written on the different possible models, methods for developing them, and their application to the design of a variety of different systems[50]. Chestnut has delineated five types of models[25] which are useful in system studies.

Model Types

1. Overall Process:
   May include schematic diagram showing hardware elements. May be functional diagram showing operation. May be a timing diagram showing sequence of operating and time of occurrence.

2. Performance:
   Accuracy - Are specifications met? Response.

3. Time:
   Scheduling of tasks, material and manpower.

4. Reliability:
   Parts. Subsystems. Total system.

5. Cost:
   Parts. Development costs.

Detailed discussion of these various models may be obtained from the reference. We are more concerned here with appending one additional model to the list. It is a composite of the performance and reliability models. This model will be called a diagnostic model. It is essentially a performance model which possesses the capability of simulating a wide range of variations in the performance of the system. It includes the specified (good) (nominal) system as a special case. Information on failure characteristics in the parts, subsystems or the overall system can be incorporated into a diagnostic model which can be solved to predict the performance at any of these levels using a model appropriate to that particular level.

Successful overall system design depends on achieving a specified performance standard coupled together with a required reliability and maintainability figure. These must be achieved for a budgeted cost. A model which is useful for evaluating the system overall in relation to these factors is termed an effectiveness model. We will now discuss this model and show in particular how it relates to the various diagnostic models.

0.2 SYSTEM EFFECTIVENESS

The quantitative measure of overall system performance is termed effectiveness[48]. This measure has been applied to assessing the performance of computing, navigation, flight control, and other hybrid* systems. Effectiveness can be quantitatively defined as the probability of the system achieving a desired objective. The objective is usually multifaceted. A greater appreciation of the role of diagnosis and its influence on the total system design problem may be acquired from observing its relationship to system effectiveness. The important constituents of effectiveness are shown in Figure 0.0. The cost of obtaining each function or attribute is illustrated in this figure by showing an arrow with notation $C_{(\cdot)}$ to emphasize that a cost is associated with each of the system constituents. A subjective measure of each cost can be stated in terms of dollars. Clearly, for most of the boxes in Figure 0.0, accurate estimates of these cost figures are often difficult to make, or for that matter, obtain, once the system has been built and is operating. Nevertheless, systems analysts attempt to assign reasonable figures to each function.

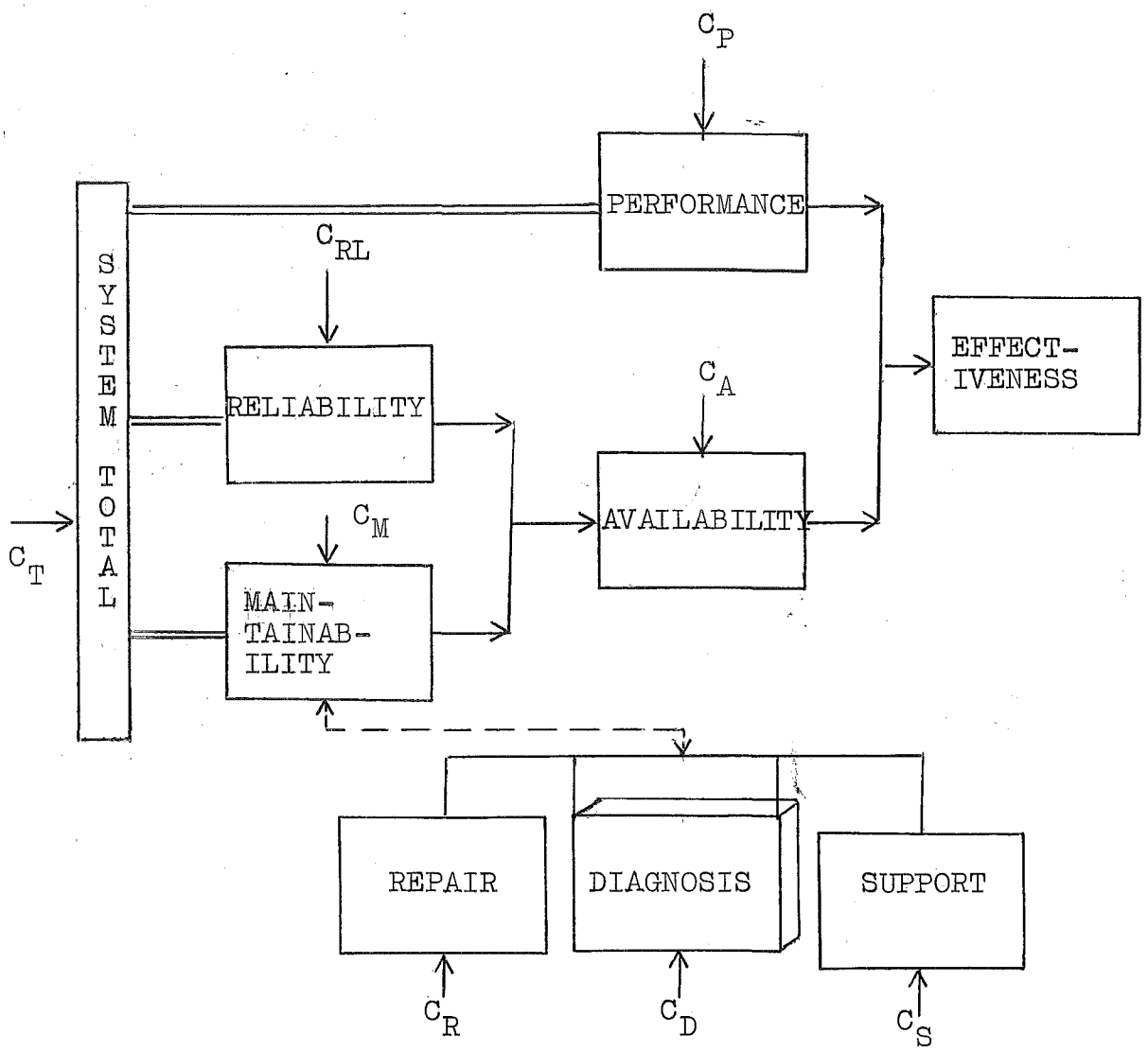
Total system cost is shown in Figure 0.0 as C_T. Often a quantity termed cost-of-effectiveness (C/E) is used as a measure for evaluating competing systems. For example, if two systems are to be evaluated, the ratio

$$\frac{\text{COST (TOTAL)}}{\text{EFFECTIVENESS}} = C_T/E$$

is computed for each. If cost is given in dollars and

---

* Hybrid is used in this instance to mean man-machine systems or systems which process both discrete and continuous information to achieve some desired function.



The components of system effectiveness and their associated costs.

Figure 0.0

If the capital for a given project is fixed, the amount apportioned to each of the four costs shown in the above equation can be varied. The objective is to obtain the maximum availability by allocating expenditure on the four areas in some optimum fashion. Because the cost of achieving a higher reliability goes up as some high order exponential, trade-off studies can be used to develop compelling arguments for fixing the reliability and seeking improvement in the maintainability function. This in turn implies a need for improved diagnostic techniques and repair methods.

Implicit in the repair function shown at the bottom left in Figure 0.0 is location of the faulty component. This requirement implies that methods for fault diagnosis must be sophisticated. They must both detect and isolate system faults and they must accomplish this with minimum human intervention and in the shortest possible time. In computing systems, this means that either hardware or software techniques or some combination of these may be used to achieve these ends. In analogue systems, both technique and specialised hardware must be developed. Finally, a cost for obtaining each of these functions must be assigned.

An idea of the saving possible from incorporating integrated diagnostics into a system can be got by considering that some of the large computing systems cost $1000 per hour. Routine maintenance takes about three hours a week and faults in the hardware can cause up to one hour per week outage. Over the period of a year, this costs roughly $200,000 and 200 hours of lost operating time.

It is apparent from this example that improved diagnostic capability in large computing systems is desirable. In systems where human life is involved, rapid

diagnosis and repair is mandatory. Depending on the function of the system - for instance, whether it is going to be used to compute aircraft landing trajectories or process airline reservations - emphasis can be placed on reliability or on maintainability.

0.3 GENERAL OBJECTIVES OF SYSTEM DIAGNOSIS

The underlying motivation for performing system diagnosis lies in the desire to increase the effectiveness of maintenance procedures and decrease maintenance time. The methods described in this thesis aim to accomplish this by providing models for developing fault information data and by evolving efficient procedures for performing fault detection and fault isolation studies.

A requirement of the methods is that they be suitable for incorporation into a computer controlled (automatic) testing scheme.

Two basic restrictions result from this requirement. One is that only a limited number of output terminals be interrogated for test purposes. The second is that tests be applied while the system is intact and that tests be interpreted electronically. This prohibits unsoldering components in the system and taking them to a bench for manual test. Electronically refers both to measurement methods and to test data processing. We are thinking here in terms of comparing the measured data with a list of pre-computed results obtained from a model. This comparison will result in a decision about the status of the system.

0.3.0 Statement of Objectives and Points of Philosophy

An aim of this thesis is to demonstrate the advantage of using analytical and digital computer simulation techniques for performing diagnostic studies. In particular, we develop models and methods in Chapters 1, 2, 3, and 4 which can be used to specify procedures for testing a system. The purpose of the tests will be to detect and isolate faults. A common theme in all chapters is the requirement for a quantitative assessment of a technique's effectiveness in detecting and/or isolating all of the system faults. The measure of effectiveness is termed testability.

An effective test should obtain the maximum amount of information about the ability of the system to perform as designed using the smallest amount of information and with minimum interference in the system operation. This latter requirement is important for a system whose continuous operation (without interruption) is essential to its effectiveness. It is generally desirable because it will eliminate or reduce testing induced faults. The requirement of minimal interference can be met in computing systems by sharing the test routines with ordinary processing if desired. In analogue systems, most test signals introduced externally will degrade performance when the system is operating. This will usually dictate that the system be shut-down during the test period. In Chapter 2 we introduce methods which permit the system to remain operational during the test application. Using the known system operating input signals and quantized-sampled output signal measurements, techniques are developed which are useful for performing fault detection and isolation studies. These techniques have not been reported previously.

Application of external input signals will not be permitted in analogue system tests. Certain portions of the internal input signal record may be selected. However, inputs may be selected for developing test information and in testing for faults in combinational and digital networks. This freedom is justified by the fact that one of the required (test) inputs will appear as one of the inputs at some time during normal operation. By observing the occurrence of the required input and the corresponding output, the desired test information can be obtained without interrupting the system. In many digital networks, and particularly in computing systems, the serial nature of the processing permits testing of circuitry which is temporarily inoperative because it is not required in the set of operations or instructions being performed by the system at that particular time.

A diagnostic model must yield accurate solutions for the physical process it is intended to model. Here, the "physical processes" will be electrical networks primarily of the analogue, combinational or sequential type. To obtain a useful system approach, we show in Chapter 1 that interface networks can be treated using diagnostic models. Actually, a hierarchy of models is developed in Chapter 1 ranging from the simplest single branch network (i.e. a single resistor, capacitor or inductor) through to a total network model which may be quite complex. In each model, the input-output behaviour must be one-to-one with its physical counterpart. This means that physical changes in components in the physical network must be mirrored by changes in the model. Thus the model accommodates to give outputs identical to the real network.

We use the word network to describe a feature of the systems being considered. This feature is the existence

of definable components. An interconnection of the components forms a network. We will affix a parameter or parameters whose value(s) can be selected to each of the network components. The physical network terminals and conductors may be considered as components. However, to simplify presentation, we usually assume ideal conductors and terminal points. The key advantage in using models to develop diagnostic information is the ability to vary parameter values to simulate corresponding fault conditions in the physical network. Furthermore, the general philosophy permits the level of model to be selected. For example, in some studies, a very fine model may be desired which has parameters corresponding to resistance, capacitance, inductance etc. values. In other applications, the level may be at the functional level. In this case, parameters corresponding to amplifier gains, summing network coefficients, logic level values etc. are required.

0.4 SYSTEM FAULTS AND THEIR REPRESENTATION IN DIAGNOSTIC MODELS

Using mathematical and simulation models, information on both fault free and faulty behaviour of a system can be developed. In applications requiring precision estimates of expected overall system behaviour, all networks within the system may be initially modelled at the finest level. Using detailed information on faults in the basic electrical network elements, a mathematical model can be formulated which simulates the precision behaviour. This information can then be used to develop coarse functional models whose parameter values can be selected to simulate the behaviour computed from the more refined model. (Skill and engineering judgment is required in developing this model.) Finally, the functional models

can be used to simulate the overall behaviour of the system for fault free and faulty conditions.

A model will be useful for diagnostic studies only if it can predict system performance both in the fault free and faulty conditions. It is, therefore, necessary to determine sources of faults, identify their characteristics, and postulate a policy for modelling or simulating their occurrence. In this section, we delineate the causes of network faults, state conditions for their electrical manifestation, and classify the faults according to their time dependence. We then proceed to formulate models which can be used to represent systems containing faults.

0.4.0 Faults

A fault is defined as the undesirable change in one or more system component electrical characteristic(s) which results in an undesirable electrical change in the system performance.

0.4.0.0 Environmental Conditions Which Cause Faults

Faults in electrical network components* are caused by inherent manufacturing flaws or by environmental conditions. We will not differentiate between causes here. It is the observable manifestation of a fault that will be of concern. Manufacturing flaws generally lead to faults which appear early in the life of a component while environmental conditions lead to faults at a later stage.

Important environmental causes of component failures are: aging, temperature, humidity, vibration and shock.

*Any definable electrical part or element. e.g. resistor, transistor, or diode.

High electric, magnetic and radiation fields can also induce faults.

Aging faults are due to changes in component material structure. Flaws introduced during the manufacturing stage may trigger gradual changes in the structure of the material. Gradual transformation in the crystal structure or variation in the chemical phase of the component material will cause the gross electrical properties of the component to change.

Temperature induced faults are the result of accelerated changes in the crystal structure or chemical phase of a section of the component material due to excessive heat. Heat may come from external sources or be generated internally due to I^2R heating within the component. Exceeding a component's power dissipation rating will lead to temperature induced faults.

Components subjected to humid or corrosive conditions will deteriorate more rapidly than under low humidity, non-corrosive conditions. Many transistor faults have been traced to moisture trapped inside the transistor can(116).

Shock and vibration cause mechanical stresses that weaken bonds on transistors and integrated circuit terminals or interconnecting conductors. Shock also affects magnetic properties of a material and will cause fracturing if the acceleration level is high enough.

Electric, magnetic and radiation fields will cause temporary impairment in component operation which may become permanent if the level of the field is great enough. Semi-conductor components are particularly vulnerable to nuclear radiation.

0.4.0.1 Microscopic Manifestation

Permanent faults and some temporary faults can be traced to a chemical, atomic or structural change in a component. We have noted earlier that faults in terminals, interconnecting conductors or in substrate or component mounting structures will not be specifically treated. The component mounting structures which may be multilayer circuit board or "cord wood" assemblies or some other packaging configuration would require analysis in detailed fault detection and isolation studies. Because they represent a particular application of techniques which will be presented, they will receive only token attention.

Microprobe analysis of failed semi-conductor structures (116) shows that flaws in crystal structure, fractures and imperfect bonding of conductors to semi-conductors cause many faults. One is tempted, in the last case, to draw the analogy between incompatible bonding problems in organic materials and organ rejection in the human body. Whether it sticks is a matter of art and to some degree, luck.

0.4.0.2 Macroscopic Manifestation

The exact manifestation of a fault at the macroscopic level will depend on the microscopic factor, on the type of component and the component signals. We define the basic lumped electrical components (resistors, capacitors, inductors and batteries) as primitive electrical network elements or primitive elements. Any components or organization of components modelled from these elements and controlled and uncontrolled ideal sources will be termed functional elements.*

* A more complete definition will be given in Chapter 1.

The existence of a fault in a primitive element or functional element can be observed as a deviation in the dependent or output signal values. This deviation is interpreted as an undesirable deviation from the normal value.

0.4.0.3 Fault Classification

Faults which exist both in the functional element and in the components within the functional elements and which are simulated by parameter settings in the model are assumed to be one of three types:

1) Degradation
2) Catastrophic
3) Intermittent.

A fault condition is very likely to lead to a second fault condition which in turn is likely to lead to a third and so on. The first fault is termed a primary fault. The faults caused by the first fault are called induced or secondary faults. This investigation will be concerned with methods for detecting and isolating primary faults.

Degradation failures - sometimes called "drift" failures - cause the input-output function of the component or functional element to deviate from specification. The component continues to function but its performance is degraded in some sense. An example of this type of fault is when a resistor drifts to a value outside its specified value (plus or minus its allowable tolerance). In all cases of this type, the component will have exceeded its specified permissible value in one direction.

Catastrophic failures result in the component or

functional element ceasing to perform in any meaningful fashion. A functional element containing a catastrophically failed part will cease to operate as intended. A catastrophic failure can manifest itself in different forms. An open circuit or short circuit condition is common in resistors and semi-conductor components.

Intermittent failures have an unpredictable effect on the functional element in which they occur. They are often due to dirty contacts, loose connections, or fractures in component material and tend to be temperature and vibration sensitive. An intermittent fault may cause operation to cease for a period of time or it may cause the functional element performance to deviate for a short time interval. Intermittent faults are usually the most difficult to handle because, although they can be detected, they are extremely difficult to isolate. They often turn into a solid catastrophic fault. Some digital computing systems are designed to tolerate certain types of intermittent faults[125, 31].

0.4.1 Diagnostic Models

Having determined the types of faults which will occur in primitive elements, how they manifest themselves electrically and their frequency of occurrence, diagnostic models of these elements can be developed to analyse the electrical behaviour of functional elements, networks and systems composed of the particular components. Faults may be regarded as changing the good element or network into a different element or network having the same structure but different component parameter values.

Often, the fault behaviour of the functional element

as a unit is of more interest than the behaviour of individual primitive elements or smaller functional elements within the particular unit. In this case, a model may be developed which does not retain a direct correspondence with the individual elements within the functional element. It may be possible to reduce the number of parameters required to simulate the fault characteristics of the physical unit. Settings of these parameters are made which correspond to changes in all or perhaps only one of the functional element components. Hence, a rather trivial but important point emerges: a particular fault in the physical network can always be represented with a setting of the model parameters but it is not always possible to uniquely define the functional element component responsible for the fault from input-output measurements made on the functional element or computations performed using the model.

To develop a useful diagnostic model, data on component reliability, operating environment and performance characteristics must be obtained. The likely classes of faults and their physical manifestation - whether they will be catastrophic, degradation or a combination - must be ascertained. Once these data have been obtained, the models for each of the subsystems can be developed. Modelling is not totally scientific; it is largely a matter of judgment and appraisal. The final model will depend on 1) the ultimate application(s); there may be several, and on 2) time and cost constraints.

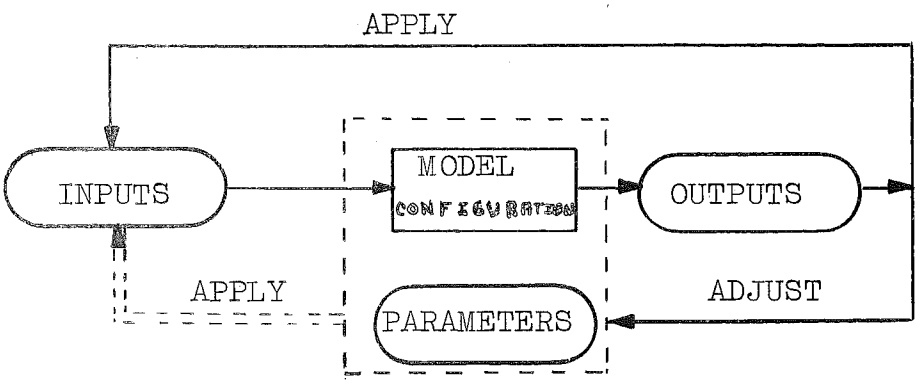
0.4.1.0 Applications of Diagnostic Models

Diagnostic models that we will be discussing have two applications. One is to develop information for devising a second model whose input-output behaviour is essentially

identical with the first but which does not retain the identity between output changes and component parameter changes. The second model may subsequently be used to develop a third model which will be more coarse in its input-ouput-parameter value relationship. This model synthesis process may be continued until the detail of the model is compatible with the required application. The second and main application is to generate test result and test procedure information using digital computer programs. This step is subsequent to the first step and depends on the models developed in the first step.

0.4.1.1 Model Synthesis

The diagnostic model formulation and development is introduced in this chapter and expanded in detail in Chapter 1. Figure 0.1 shows the basic formulation-development procedure. A model configuration is specified



MODEL FORMULATION
Figure 0.1

and parameters are assigned. An input is applied and the output is obtained. A next input may be applied (top loop) or a parameter may be adjusted (bottom loop). Eventually, by making model parameter adjustments, a diagnostic model results which gives the desired result.

The set of diagnostic models which can be developed forms a spectrum which ranges from the detailed analytical model which is one-to-one with the actual system to the coarse functional model, often called a "black-box" model. The former will be referred to as an isomorph and the latter as an homomorph. Simulation models may be of either type.

0.4.1.2 Fault Simulation Policy

A diagnostic model is useful for predicting performance characteristic of a functional element or network for a limited class of faults. The model is limited in both the number and type of faults it will represent. This limitation will be termed the model fault policy. The fault policy selected depends on the actual component characteristics, including the likely occurrence and physical manifestation of a fault, and on limitations imposed by time, ingenuity and budgeted cost for the system diagnosis studies.

The fault policy assumed in this study can be summarized as follows:

1) Faults are assumed to occur one-at-a-time.

2) Faults may be of the catastrophic or degradation types. In some cases, the distinction between a degradation and catastrophic fault is not necessary, catastrophic faults being a particular case of degradation faults.

3) Faults will occur with greater likelihood in some components than in others. Faults which result in structural changes in the network or in terminals or interconnecting wires are assumed to be less likely than internal component faults. Consequently,

even though there is no inherent limitation in modelling these faults, they will not be treated explicitly.

We now consider the general model formulation and evaluation process.

0.4.1.3 Model Formulation and Evaluation

Figure 0.2 shows a diagram of the model building and evaluation process used for diagnostic modelling. The first step is to devise a parameter dependent model. This model must mimic the physical component or network in terms of input-output performance. Characteristics are verified in the second step. If the characteristics are not identical, the model must be revised. At this point, a problem is encountered: One must decide either to adjust parameter values or to change the form of the model. This change is largely a matter of judgment. In most cases, and in particular for standard electrical circuits, the correct model form can be obtained from physical or functional knowledge of the component or network. In large networks this may not be so easy.

The model is revised either in form or by altering parameter values until a solution is found which matches that of the physical network. The third step involves a performance comparison between the network and its model.

We digress momentarily to reiterate a previous point. It was stated that the model must be one-to-one with the network in terms of its input-output behaviour. It is also desirable that the model parameter values be one-to-one with the component parameter values in the network. If this condition is satisfied then a change in the

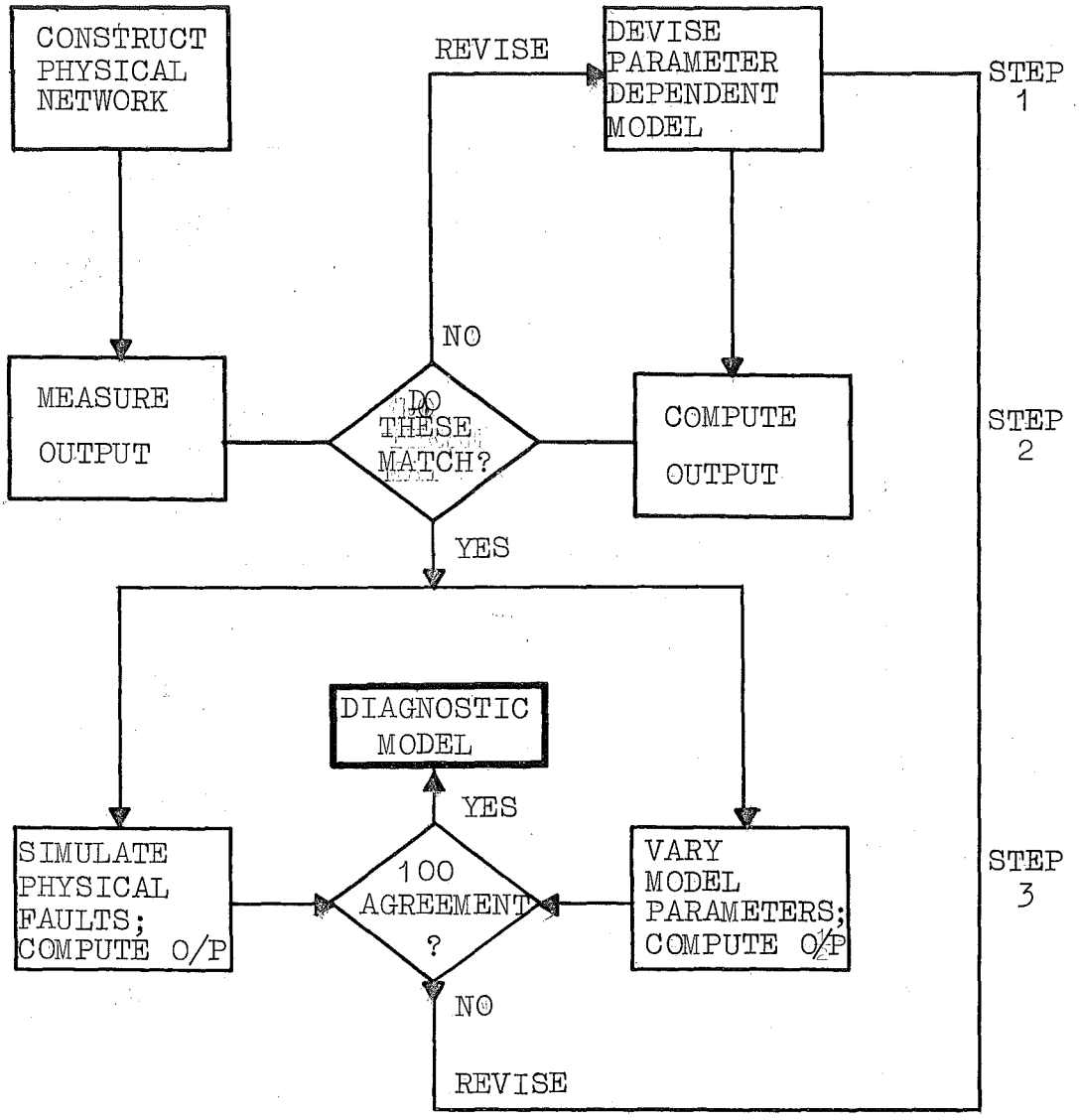


Diagram showing the diagnostic formulation and evaluation process. The final diagnostic model should be accurate in all detail. The comparison with the physical simulation is not always possible.

Figure 0.2

parameter value in both the network and model will result in identical input-output behaviour for both the network and its model. What we are saying is that if possible, an analytical model of the network should be developed which represents all solutions to the network. Solutions obtained from the model will be identical to solutions obtained from the actual network. However, in a functional model, detail of the effect of individual components is disregarded. A "black-box" parameter (e.g. gain, impedance etc.) may be specified to describe the input-output operation. Certainly each internal component will influence the "black-box" parameter. But the difficult question to answer always is; which, how and how much?

Continuing with the third step, we note in Figure 0.2 that the one-to-one model will usually give a match and result in a useful diagnostic model. We will want to use models which, for reasons of economy, do not contain the fine detail of the one-to-one model. In this case, we will postulate a subset of physical network failures which the model must simulate. It is generally easier to devise models which simulate a small number of failures rather than the entire spectrum. However, as we shall see in Chapters 3 and 4, in digital networks, a limited number of failure modes in the model can be used to simulate a wide variety of errors in the network. In this case, a correspondence between physical and model parameters does not exist.

0.5 ANALYTICAL DIAGNOSIS: TEST INFORMATION AND PROCEDURE DEVELOPMENT

Analytical diagnosis is essentially the development of system diagnostic models and their application to

test information development and testing procedure specification. Models are developed to simulate a class of faults dictated by the fault policy and solutions for particular inputs and particular fault conditions are obtained. These solutions are used to develop testing procedures for the system. Input-fault-output information obtained from the analytical model can also be used to compile maintenance aids such as fault dictionaries, troubleshooting manuals and to provide fault data for computer controlled fault detection and isolation equipment.

In this thesis, data generation and procedure development are oriented to digital computing techniques. That is, the model solutions will be obtained using programmed algorithms. A distinguishing feature of methods for analytical diagnosis developed in this investigation is the use of discrete samples of the model output signals. This is expected for digital systems. Because most testing methods for analogue equipment and in particular, linear analogue equipment are based on transfer function or correlation[180], they usually use continuous signal in formation. These methods are effective but have the objectionable requirement that input signals be applied.

In Chapter 2, methods based on continuous input, sampled-quantized output information are developed for analogue networks. The advantage of this approach over the conventional approach is that all digital measurement and evaluation techniques can be used in the system and no external signals need be applied. This reduces the test data storage requirements and simplifies the test execution and result evaluation problem. Chapters 3 and 4

discuss corresponding model applications to combinational and sequential networks respectively.

Analytical diagnostic studies can be justified only if they improve the efficiency and effectiveness of fault detection and isolation in the system over that of the empirical methods. Empirical methods cannot, in general, be quantitatively evaluated for efficiency and effectiveness. A major advantage of employing analytical studies is that they do give quantitative a priori estimates of the efficiency and effectiveness expected from a particular method when applied to a particular hybrid system.

0.6 APPLIED DIAGNOSIS: SYSTEM CHECKOUT EQUIPMENT AND PROCEDURES

System checkout, which will sometimes be referred to here as system testing, hardware testing or some equivalent term, is an activity which is directly concerned with applying signals and making measurements to answer one or all of the following questions:

a) Is the system operating as designed?

b) Is there a fault in the system?

c) If there is a fault, where is it located?

System checkout is performed during the time when the system is being prepared for delivery, during the period while it is being installed, and during its operating life.

System diagnosis, which includes the system checkout activity, will be tentatively defined here as that part of the systems discipline concerned with developing models, prescribing algorithms and performing computations

which are useful for

(a) Defining tests

(b) Developing test procedures

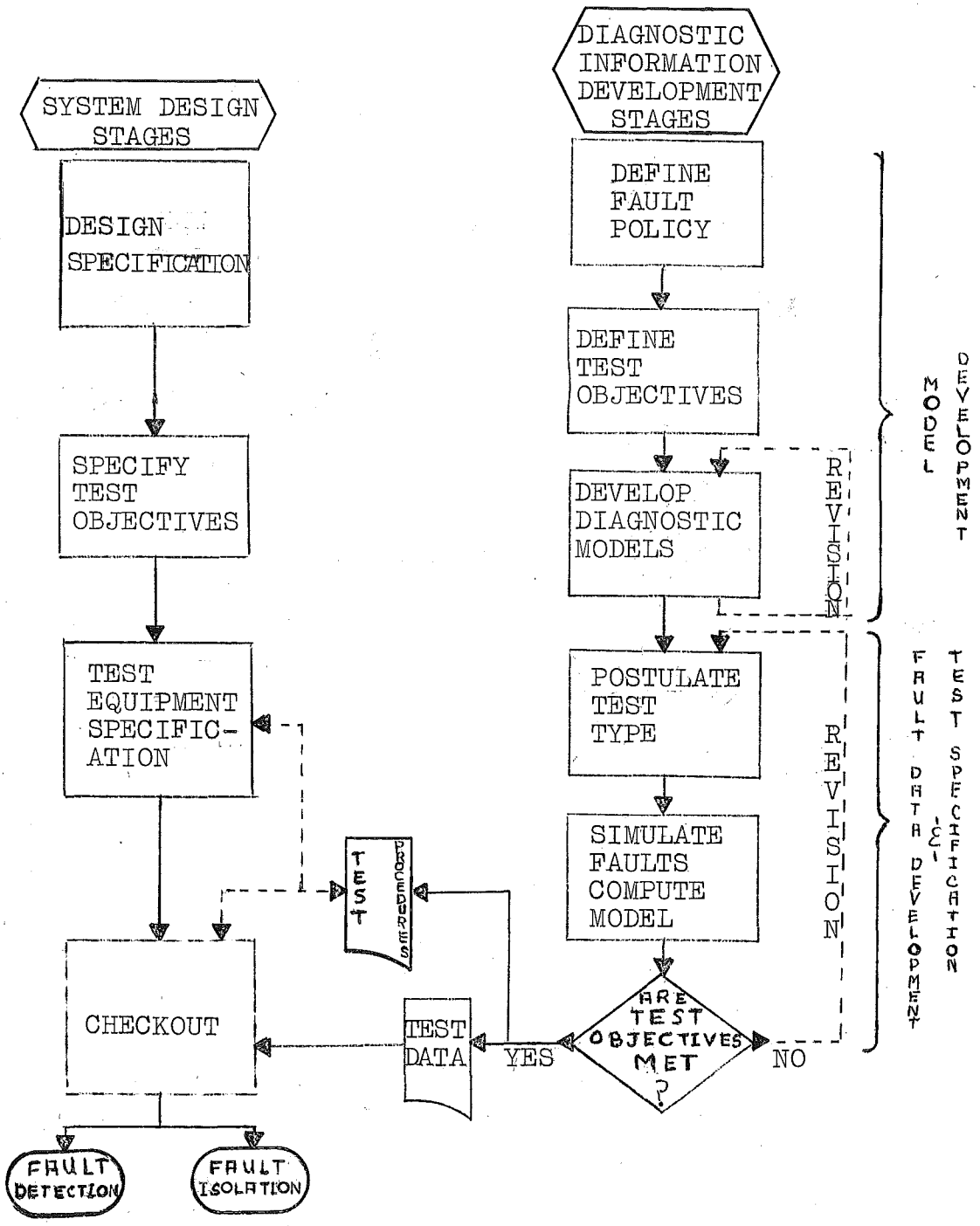
(c) Developing test data

The activity of performing a system test is guided by (b) and the analysis of test data is closely tied to (c). The overlap between the theoretical and applied sides of this area of the systems problem is necessary and desirable. The interdependence is typical of engineering analysis and design. In this case, the overlap ensures the maximum interaction between the system performance design and the system test design personnel. This interaction is important for high system effectiveness.

0.6.0 A Priori and A Posteriori Diagnosis

The development-application concept of diagnosis consists of two stages. The first stage is basically what we have defined as system fault analysis. The second stage is essentially system checkout. But because checkout includes equipment, measurement devices and personnel, we have selected the term a priori to denote those diagnostic activities that can be performed analytically, computationally, or algorithmically before any of the system hardware takes shape. Those activities which apply this "pre-hardware" derived information are termed a posteriori diagnosis. Using these terms, we separate and identify those aspects of the system checkout problem which are more related to the analytical side than they are to the hardware development side.

Figure 0.3 shows a simplified breakdown of the system design stages and the diagnostic information development



Block Diagram relating system design to diagnosis.

Figure 0.3

stages. The a priori diagnostic stages are divided into two areas. One is the Model Development area and the other is the Fault Data Development and Test Specification area. These activities have corresponding functions in the system design areas. This correspondence is indicated on the diagram by grouping corresponding functions in both the system design and diagnostic information at roughly the same level in the diagram. The diagram indicates "sphere of influence" relationship that exists between the two activities moving vertically. The a posteriori - a priori relationship between the system and diagnostic information development phases can be imagined by moving horizontally at the Fault Data Development and Test Specification Levels.

The a posteriori side of the problem of system diagnosis can be guided by results from the a priori side. However, verification that the software generated data and procedural techniques give reliable and useful results can be obtained only after the system is designed and operating.

During the system design stage, the diagnostic model development stage must be carefully evaluated for accuracy and authenticity. If, for example, the set of faults postulated in the fault policy is not representative of the type that will occur in the actual system, then no matter how good the model and its ability to simulate the selected fault conditions, unless they correspond to what may happen in the actual system, the method developed and data obtained will be next to useless. An updating and revision procedure in Figure 0.3 is indicated by a loop in the diagram to emphasize that this stage may require revision as data on likely faults and their manifestation in the actual system

becomes available. In a sense, the fault list may grow with the development of the system. This in turn requires an updating of the test specification routine if the particular fault has not been previously postulated in the model. This activity is shown on the diagram in Figure 0.3 as Model Development.

The second major area of activity in the a priori diagnostic problem is that of Fault Data Development and Test Specification as shown in Figure 0.3. It involves a suitable model, the specification of a set of input signals, a policy for simulating the faults, the computed outputs and a decision on the pertinence of the output with respect to its ability to (a) indicate the presence of the particular fault and (b) give a unique indication for the particular fault. Both of these indications will depend on the input applied and the output terminals and the length of the output record. For time independent networks such as D.C. networks and combinational digital networks, the record length will have little bearing on the test specification. In networks containing memory, the record length is very important. These points will be clarified in later Chapters. (For example, see Chapter 2, Section 2.9.) The requirements of the test specification stage are complex. In Figure 0.3, a loop is shown as linking the decision block which is the step of the process during which the test is evaluated back to the test type postulation stage. In reality, the activity will be much more complex than indicated on this diagram. There are in fact, multiple loops that correspond to evaluating different outputs and inputs for each type of fault simulated and then the selection of a set which is in some way optimal. Measures will be described in Chapter 1 which permit quantitative estimates to be made of the effectiveness of a particular test procedure or

even of a single input-ouput pair to detect and/or isolate a class of faults.

0.7 SUMMARY AND CONCLUSIONS

In this chapter, diagnosis has been related to the general systems engineering design problem. The relationship is established by developing system diagnostic models and outlining the application of these models to obtaining information which is useful for performing system checkout at various levels. We have emphasized checkout at the operational level.

It has been stated that the types of system components (sub-systems) which will be treated are: analogue, combinational, sequential and interfacing networks such as comparators or ADC's etc. In using the word network rather than retaining the word system, we are tacitly assuming certain functional and structural properties. Important among these are that the system can be sub-divided into definable units, that these units can be modelled, and that the models are network models. A network model does not admit distributed parameter systems. Consequently, transistors must be modelled by lumped equivalent functional models. Components within sub-systems must be lumped and uniquely defined.

The modelling procedure outlined illustrates the basic diagnostic model development scheme. It is essential for diagnosis, to develop models which accurately simulate faults. The causes of faults and their manifestations were discussed briefly. The interpretation of a fault in terms of a model parameter variation establishes the final link between the physical world and its model counterpart.

Although a one-at-a-time fault policy will be imposed in the remainder of this thesis, it should be emphasized that it is not an inherent limitation of the models. They can be used for multiple fault simulations. The limitation is rather one of expediency and economy. Take for example a functional model having 10 parameters whose values come from the set of integers to 1000. If faults correspond to one half of the possible combinations and output solutions are required to obtain isolation data for these cases, about 2^{90} solutions will be required. This figure is truly astronomical. A reasonable compromise would be to assume two-at-a-time faults. However, the advantage of this over one-at-a-time seems marginal but may be justified in certain conditions.

1 DIAGNOSIS

Discovery consists of seeing
What everybody has seen
And thinking what nobody has thought

Albert Szent-Györgys

1.0 INTRODUCTION

The word diagnosis varies in its meaning, depending on the discipline within which it is applied. In general, diagnosis is an activity directed toward answering such questions as: "What is wrong?", "What is the ailment?", or "Does this symptom indicate that cause?".

To perform diagnosis, information pertinent to the functioning of the object being diagnosed must be collected. Data, once gathered must then be processed and interpreted. Finally a diagnostic decision must be given. The confidence in a diagnostic decision depends on such factors as total information gathered, reliability of the information, knowledge of the object being diagnosed, and the diagnostician's experience and training and general ability to select relevant information from the irrelevant information.

It is sometimes argued that the system diagnosis problem is analogous to the medical diagnosis problem. For example, a system fault of malfunction corresponds to a disease state in a human being. Similarly, gathering information on the system functioning corresponds to

developing and applying medical tests. Even the word symptom carries similar connotation in both cases. However, the analogy should not be extended much further than this; the main reason being that medical diagnosis at its present stage of development is less analytical than system diagnosis can be.

There are several factors which make the problem of system diagnosis more tractable than the corresponding medical problem. We will mention two. First, system models can be constructed into which states of disrepair can be introduced. This is not generally done on human subjects. Second it is possible to access almost any point within a system for measurement purposes. The medical diagnostician is usually more limited in this respect than the electrical hardware system designer. Considering these two factors and others mentioned in Chapter 0, we can state some additional points of philosophy behind this investigation and the factors motivating it.

It is possible to diagnose system equipment which has been designed by applying signals to it, measuring its output signal response, and by subsequently computing various characteristics from these data. Possibly, the measurements and data will be sufficient to indicate the presence of a fault. Again, having observed a malfunction symptom, perhaps as a deviation in some output signal from its design value, another series of measurements can be used which will hopefully obtain data sufficient for isolating the malfunctioning component.

The words "possibly", "perhaps" and "hopefully" are used in the previous paragraph to emphasize a point; without a thorough and detailed analysis of the system being diagnosed, one can never be certain that test

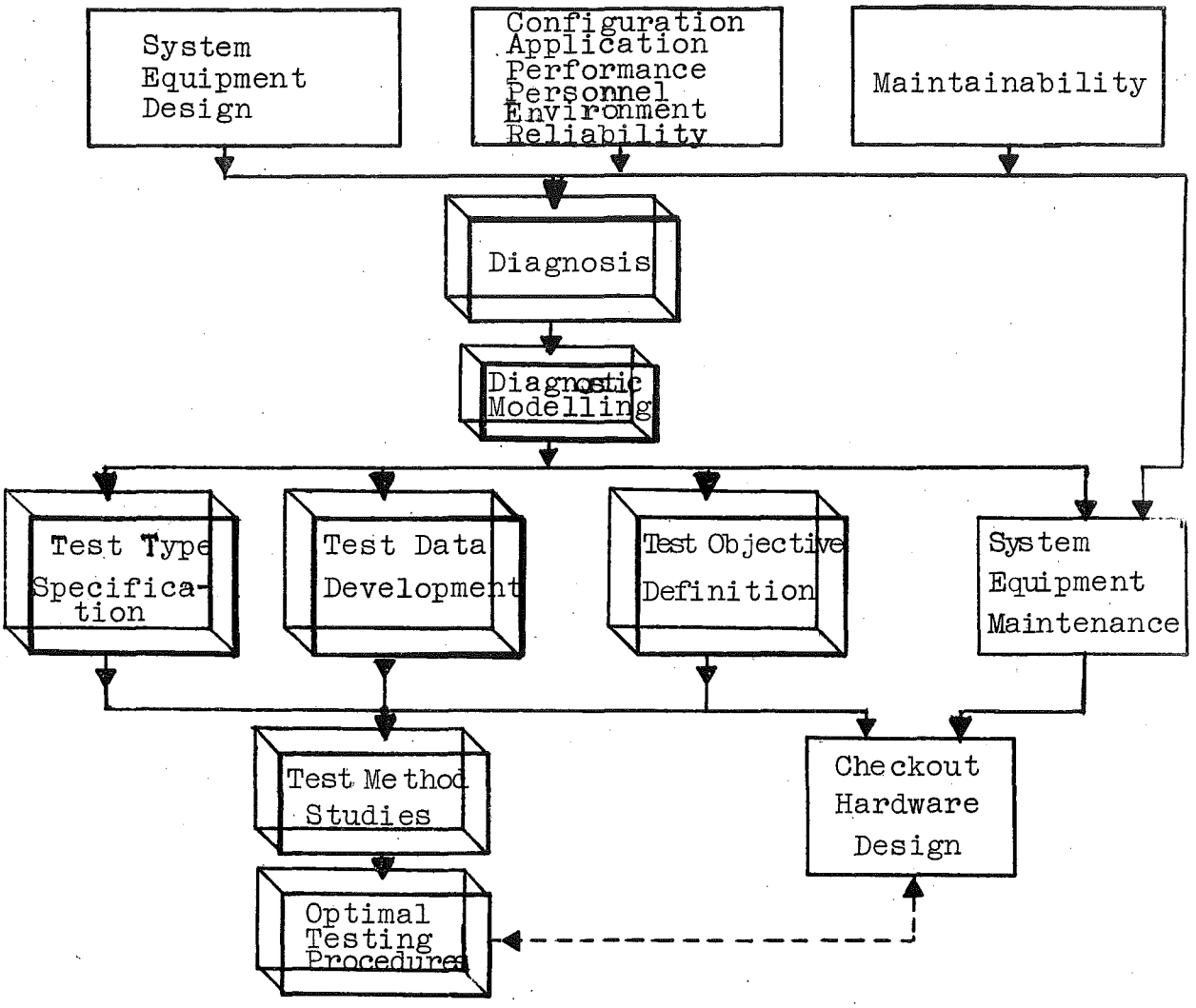
information obtained from the system will be relevant. Even with a careful and detailed a priori fault analysis, there will be some doubt or lack of confidence in the interpretation of the test information. But, there is good reason to believe that carefully conceived methods and thoroughly analysed test data will permit rapid and reliable testing procedures to be developed which give high confidence results.

Two points are worth emphasis. First, for large systems in which computer controlled testing is applied, it is necessary to have a priori information on signals both for the fault free and faulty system. Second, in all tests using computer controlled checkout equipment only a limited number of signal terminals is available for measurement. Consequently, malfunction isolation must be accomplished indirectly using "black box" measurement information. This is the antithesis of the repairman or trouble shooting philosophy which permits access to almost any of the terminals for measurement purposes and which allows unsoldering and replacement of components for test purposes.

Figure 1.0 shows an organisation chart of the diagnostic studies carried out in this investigation and described in this Chapter. In Chapter 0, it was shown that diagnosis is a part of the maintainability area. In practice and as we have illustrated in Figure 1.0, diagnosis can depend on equipment maintenance*, (EM) constraints. Although not specifically detailed on the

---

*From here on equipment will be used to refer to the physical electrical system. The word system will be used to mean the equipment model or the actual equipment when no ambiguity exists.



ORGANIZATION OF DIAGNOSIS

Figure 1.0

chart, the dependence is bilateral. There are a large number of factors that have been grouped together in the block in Figure 1.0 labelled configuration, application, performance, personnel, environment, reliability (CAPPER). These are some of the important factors - both human and technical - influencing diagnosis. The scope of system diagnosis can be observed by noting the number of functions for which diagnosis is responsible. Diagnostic modelling (DM) is shown as having an effect on the system equipment maintenance (SEM) function and the checkout hardware design (CHD) function. Various test development functions specified in Figure 1.0 are: test type specification (TTS), test data development (TDD), test objective definition (TOD), test method studies (TMS) and optimal testing procedures (OTP). Horizontal dependence among functions at the same level in the organization is implied.

1.1 CHAPTER OUTLINE

The purpose of this chapter, indeed this thesis, is to develop the theme of a priori diagnosis by elucidating features and interrelationships of those diagnostic functions shown in blocks with shaded borders in Figure 1.0. In this chapter, the definitions and assumptions required to develop diagnosis according to the organization in Figure 1.0 are stated and related formulae are derived. Diagnostic modelling which was introduced in Chapter 0 is extended and made sufficiently explicit for developing models which can be used for TDD, TTS, TMS and OTP studies.

We begin in Section 1.2 by surveying some historical developments in diagnosis pertinent to the systems approach. A statement of the research objectives is then given in Section 1.3. In Section 1.4 some useful definitions which were not previously presented in the Prologue or in Chapter 0 are stated and some of the

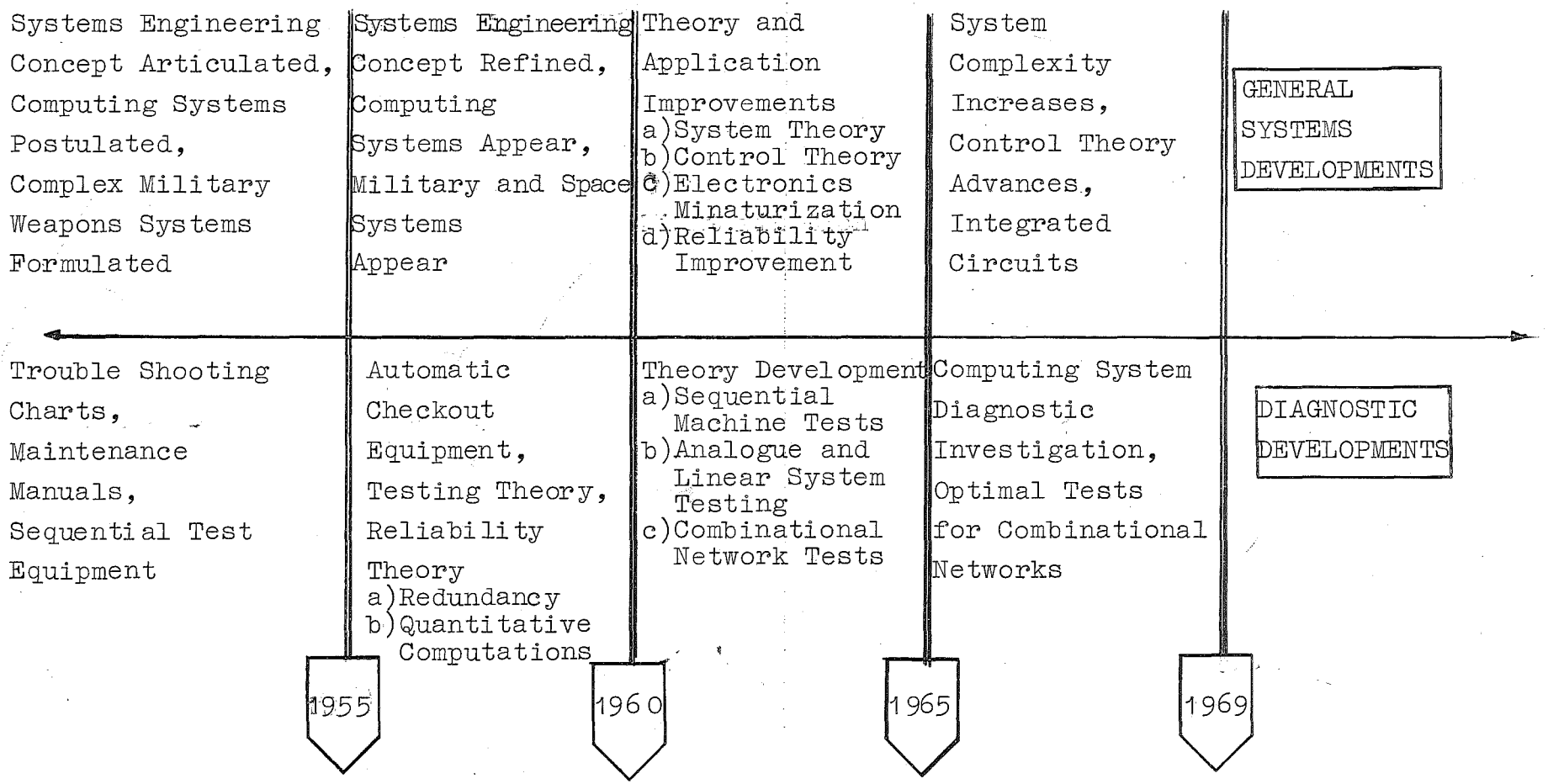
definitions already mentioned informally or used in a general context are given. After presenting an example in Section 1.5 which illustrates limitations on obtaining diagnostic information and cites some problems which are encountered in developing analytical techniques, we develop the basic hybrid system component models that can be used to model hybrid systems' networks for diagnostic studies in Section 1.6 then in Section 1.7, the test development areas specified in Figure 1.0 are discussed. Finally in this chapter in Section 1.8, an example of a diagnostic data generation and fault detection-isolation illustration is presented using functional element network fault simulation techniques.

1.2 DEVELOPMENT OF HYBRID SYSTEM DIAGNOSIS

1.2.0 History

The background material presented in Section 0.1 of Chapter 1 presented important events affecting systems engineering development. It was postulated that system analysis through modelling is of significance to the area of systems diagnosis. The development of hybrid system diagnosis has been largely evolutionary. A temporal description in Figure 1.1 shows a few of the important events both in the general systems development area and in diagnostic developments. (The remainder of this section is with reference to Figure 1.1.)

Although large systems (e.g. power systems, transportation, logistics) existed before 1950, the invention of the transistor and the increasing emphasis on more complex military and computing systems precipitated the general systems engineering concept. Figure 1.1 shows that prior to 1955, diagnosis was usually



A Temporal Comparison of Systems Development and Diagnostic Development

Figure 1.1

an afterthought of the system designer. The general approach was to supply "trouble shooting" charts and repair manuals developed after the system had been designed. Nevertheless, in some systems, the first automatic test equipment began to appear.

It was around 1955 that John von Neumann, E.F. Moore and C.E. Shannon published papers on the use of redundancy techniques to improve system reliability. They showed that by using redundancy techniques, systems containing electrical components with very short mean-time-between-faults (MTBF) could be made to work over periods of time much longer than the MTBF time for individual components. However, cost, weight, volume and power sizzling requirements precluded general application of these techniques at that time.

Between 1955 to 1960, system engineering concepts were sharpened and complex missile, satellite and space systems made their debut. Many advances in technological application of previous inventions occurred. Reliability and quality control theory advanced. 1960 to 1965 was a period during which systems theory, control theory and reliability studies gained much attention. The demands of complex space problems and industrial control problems being the impetus behind this development. At about this time, diagnosis acquired respectability as an important part of the systems engineering problem and theoretical studies on diagnostic problems made their debut.(147)

In many fields, for example circuit theory, the evolution and organization of developments is fairly easy to follow. In diagnosis, there has been no clear picture of the organization or even of the objectives except in one or two areas. One main theme that can be

followed is the development of automatic checkout equipment and its applications in avionics and space system. Power system protection could also be included here. Another is the concerted effort by computing systems manufacturers to develop rational circuit oriented diagnostic techniques. Some of the important early references on systems diagnosis are contained in unpublished company reports. These have appeared gradually several years after being promulgated within the company.

1.2.1 Literature

A paper by E.F. Moore on sequential machine experiments might be considered as the first reference on the subject of system diagnosis.[107] Although Moore deals with finite state machines having a finite number of input and output signals and a finite number of states, his concept of an experiment is general enough to be applied to models other than those for which it was originally postulated. For example, it should be possible to extend the concept of an experiment approximately to essentially infinite state machines by quantization of the continuous state variables. Moore mentions other possibilities in his paper.

The advantages of formulating a diagnostic approach which can be applied to more than one specific system class was first enunciated by Brule, Johnson and Kletsky[19] in 1960. Their treatment of TTP and OTP is general and based on a system description in terms of an interconnection of sub-systems* (sub-assemblies).

---

* Sub-assemblies are equivalent to a network or functional element.

Signals are transferred through terminals leading into and out of these sub-assemblies. They state that a test which will verify that the system is good must give a complete specification of the inputs to each sub-assembly which are required to generate the specified outputs. Only if each sub-assembly is operating properly is it possible to state that the system is good.

In developing test procedures, Brule et.al. assume that a test will "pass" if all sub-assemblies are good and fail if any one (or more) is bad. An example illustrating the important features of their test procedures is shown in Figure 1.2.

The example system illustrated is composed of the interconnection of three sub-assemblies a,b, and c. The primary (independent) stimuli are shown as S_1, S_2 and S_3. Intermediate signals and output signals are denoted by capital letters A,B, and C. The table in Figure 1.2 shows the response signal obtained from the application of certain stimuli. We note here, that intermediate signals can be regarded as stimulants which are assumed to be good. (Brule et.al. use the term stimulus - response to stand for the input-output signal responses of a sub-assembly.) For example, the table shows that the stimulus S,A will give the required response B only if the sub-assembly b is good. A coding of the status of the sub-assembly, necessary for a test to pass, is given by the column in the table labelled Numerical Designation. If sub-assemblies are alphabetically ordered from left to right then a 1 is entered in the column if the sub-assembly must be good for the test to pass. If the sub-assembly is not tested by the test a 0 is entered in the corresponding column. For sub-assembly b, the

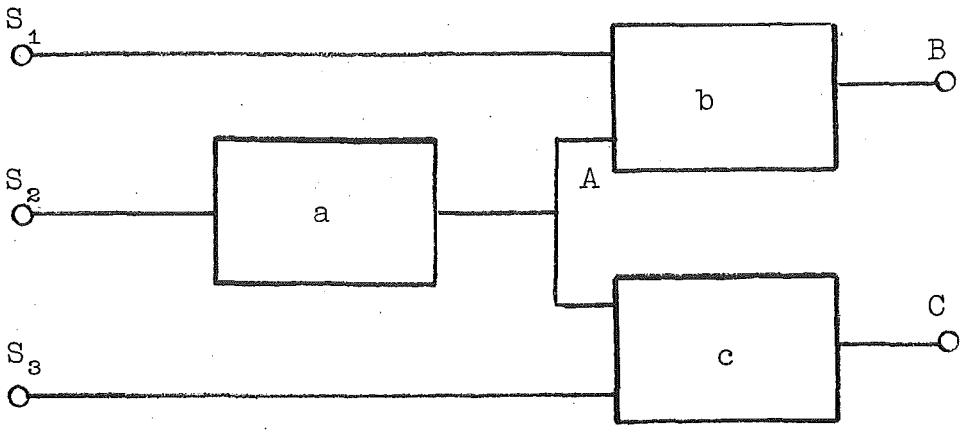


Diagram of System Showing Sub-assemblies

Row Number	Response Observed	Stimulants Required	Good Sub-assemblies to Pass	Numerical Designation
1	A	S_2	a	100
2	B	S_1A	b	010
3	C	S_3A	c	001
4	B	S_1S_2	ab	110
5	C	S_2S_3	ac	011

LEGEND: 1 = element tested
0 = element not tested

Equipment Diagram and Test Requirements
Figure 1.2

test passing gives 100. In the first row of the table, the response A, given stimulus S_2 is good if sub-assembly a is good, independent of the status of b and c. For the system shown - considering primary stimulus signals only - the two input pairs S_1S_2 and S_2S_3 and the response signals B and C are required to verify that a,b and c are operating properly.

Slightly more specific systems oriented diagnostic approaches have been developed. One, reported by Maling and Allen[87] uses a large simulator to develop tests for an experimental central processor in a digital computing system. Hardie and Suhocki[52] have reported another simulator which was applied to a special purpose flight control computer. Their simulator is interesting because of the number of fault types it can handle. An investigation using physical fault simulation in a telephone switching network was carried out by Tsiang and Ulrich[137] at Bell Telephone Laboratories. They used an actual telephone exchange and compiled a fault dictionary of sympton versus likely cause(s) using physical fault simulator techniques. This data was later analysed by Kruskal and Hart[78] who devised a method for compressing the data information using a Hamming distance defined on the data space. This compression resulted in a shorter fault dictionary than the original one of Tsiang and Ulrich.

Work by Seshu and Freeman[128] and Seshu[126] are frequently cited references on digital system diagnostic development. Their method is essentially a simulation algorithm designed for finding input-output patterns for both good and faulty networks and is based on the following assumptions: 1) the class of failures which can occur is known and finite, 2) each failure transforms

a sequential network into another sequential network and 3) it is possible to reset feedback lines momentarily to a known initial state even under failure conditions. The good network and all faulty networks (machines) are simulated. During each step of the simulation, all possible combinations of inputs are tried and the one which detects the most previously undetected faults is used as the next input combination. Manning [91] who studied under Seshu uses this technique to investigate procedures for rapid and efficient fault detection and isolation on the OSX-1 experimental computer which was built at the University of Illinois.

Important contributions to diagnosis in the analogue systems area are few in number. One due to Berkowitz [13] discusses the requirements for finding R,L, and C values for linear passive networks using limited input-output signal information. Seshu and Waxman [127] use sine wave input-output information to develop fault detection and isolation information for linear networks using a transfer function approach. Valstar [141] describes a technique which permits on line continuous fault detection and isolation for linear networks using a transfer function tracking scheme and digital computation while Levadi [81] describes a technique which applies a learning algorithm and which can be used for linear or non-linear networks. The FIST [129] project was initiated to develop test equipment for modularized electronic equipment. The resulting technique is based on the use of two part comparator circuits. The philosophy is that if equipment is constructed in modules and fault detection tests are performed at the replaceable module level, then rapid repair is possible once a fault is detected (provided the replacement module is available).

Contributions to diagnostic techniques for combinational networks are more numerous than for any of the other network types. There are two reasons for this. One is the fact that many of the circuits in digital computers are combinational. The second is that the problem of diagnosis in combinational networks gives the impression of being more tractable than the analogue or sequential network, an impression which is somewhat misleading. Various OTP's have been proposed but their approach to the solution is usually quite different. For example, one approach due to Poage[115] is an algebraic method which modifies the Boolean equations for the network to account for faults on signal lines. The modified equations are then manipulated and an expression for the network output for various input signal combinations and for possible faults is obtained. A second approach for diagnosing combinational network faults has been developed by Roth[122] to a sophisticated level over a period of several years. He bases his work on an algorithmic procedure which he calls D-cube calculus[40]. A reduction procedure incorporating multiple intersection of appropriate truth table like blocks which give a complete description of each gate and finally the total network is used. Some of the IBM 360 diagnostic programs have been designed using his algorithm in its programmed form, DALG.

Some of the work on sequential network diagnostic techniques, for example, that of Seshu and Manning mentioned previously, has been carried out assuming that the network exists within the computing system environment. This assumption is, from our point of view, a useful one. Because the sequential networks form a part of the total hybrid system, previous methods

for diagnosis which consider the network imbedded in the system are relevant to this study.

It has already been stated that the work of Moore on sequential machine experiments is of interest because of his treatment of an experimental test result is general enough to apply to a wide variety of situations. This subject area has received further treatment by Hennie[56] in 1964, by Kime[71] in 1966 and Kohavi and Lavallee[72] in 1967. Although the work of these investigators has application to fault detection and to sequential network design, it has not yet been applied to fault isolation.

Limitations which apply either in part or in entirely to most of the diagnostic methods or techniques reported in the literature can he summed up as follows:

(a) The methods yield solutions or prescribe techniques which are not viable in the system environment. In particular, most of the analogue techniques require sinusoidal input signals which may not be available in the operating environment of the system. Others require access to a large number of test point terminals.

(b) The effectiveness of the method in detecting and/or isolating faults cannot be quantitatively evaluated.

(c) The time required to develop diagnostic information using a given algorithm or heuristic may be excessive.

Admission of these basic limitations is not enough. It is the authors contention that a more rational approach to the problem of diagnosis considered within the systems engineering context is required. This approach can be implemented by

1. Specifying the important constituents of an organized diagnostic framework.

2. Relating past and future contributions to the corresponding constituents of the framework selected.

3. Developing both qualitative and quantitative measures for analysing and evaluating the significance or effectiveness of a particular contribution.

1.3 OBJECTIVES OF INVESTIGATION

1.3.0 Specification of Objectives

The major objectives of this thesis are: a) to present a philosophy for system diagnosis, b) to explicate the philosophy by developing definitions, terminology, and models which provide a framework within which diagnostic studies on a large class of electrical systems may be performed and c) to demonstrate how the general conceptual framework can be used to develop test data and specify checkout procedure requirements for hybrid systems operating in a real time environment. The test data and procedural information development and application are digital computer dependent. Because the systems of interest in this investigation are known to be complex, manual testing is ruled out, time limitations on tests being a critical factor. Present computing systems have special programs and circuitry for diagnostic purposes.(125) By incorporating analogue-to-digital converters and electronic commutators, it is a short step to making test measurements in the integrated hybrid system. These measurements are useful only if a priori computed information is available. The measured and computed information can be compared and a decision about the status of that section of the equipment can then be made.

The a priori computed information, applied a posteriori, can be developed using diagnostic models incorporated into a digital computer simulation. Algorithms programmed for the digital computer can be used to select input terminals, input signals, output terminals and to compute output signals for various fault conditions. This computed information can then be used to develop checkout equipment, test sequencing (test procedures) and it can be stored on magnetic storage media for analysing future a posteriori fault detection and isolation decisions. (See Figure 0.3)

The stress that is placed on the digital computer as a computational tool and as a measurement control and information processing device is important. It extends system diagnosis to the third dimension; previously combinational and sequential circuitry in a computing system could be treated. Now, the hybrid system can be considered as a unit.

1.3.1 Discussion of Objectives

An effective system design must consider diagnosis as one of the design areas. This implies that systems analysis must be oriented to the need for relating analytical studies to equipment design. The process of integrating the diagnostic studies with checkout equipment can best be done using a rational approach. The rationality can be acquired from a well defined organization of diagnostic functions. The individual contribution of any given technique can then be assessed within the context of the particular organization.

We reiterate that implementation of diagnosis in the form of computer controlled automatic test sequencing and test analysis equipment is the ultimate objective

of diagnostic studies. Intermediate objectives are the substance of this study. To attain these objectives, the organization established in Figure 1.0 will be further qualified by providing definitions "as you go" for DM, TTS, TDD, and TOD functions. Using this organization, and the resulting definitions, answers to questions of the following general type can be given: What model will be useful for developing fault detection information data on an integrated circuit operational amplifier assuming faults are caused by parameter value drift? Which subclass of a given class of faults can be detected using a particular input signal record and the corresponding output signal record for a combinational digital network?

The organization hierarchy depicted in Figure 1.0 shows that TMS can be investigated using information from DM, TTS, TDD and TOD. OPT depends on information from the TMS area. However, the CHD and OTP functions are buffered by equipment performance and maintenance constraints. A sort of dialogue between these two must be established in practice which uses "advice" and information from the other functions in the organization.

We assert that, for present generation systems and those in the planning stages which involve human life and other costly resources, system effectiveness must be measurable and maximised. The effectiveness of tests and test procedures, be they for routine (scheduled) maintenance or for fault (unscheduled) maintenance, must be a priori assessed for thoroughness or completeness. If, for example, a particular black-box test technique will detect only 90 of the possible 100 assumed faults and high system effectiveness is mandatory, provision for redundancy or additional test equipment

hardware to cope with the remaining 10% must be supplied. The measure of thoroughness which we adopt here is termed diagnosability or testability.(See 147) Efficiency is measured on a relative scale and depends on cost, time and other physical constraints.

If the models of basic system components accurately and reliably predict the performance characteristics of the physical system component for both fault free and faulty conditions, the model is useful for diagnosis. We will assume that for a postulated class of faults, the basic model is 100% effective in its ability to predict all good and bad modes of performance.

The process of modelling or model building may be imagined to be evolutionary; starting with basic components, increasingly complex systems are constructed by interconnecting the basic components. Partly for convenience but mainly because simplification is highly desirable, certain important interconnections (sub-systems or networks) are identified and these become new "basic components" for larger systems and so the process continues. At each stage of the evolutionary modelling process, several factors should be assessed. These are: 1) How close a resemblance does the model bear to its physical counterpart? 2) Can the model simulate a class of faults which the physical component (system) may possess? 3) Is the overall performance of the model both for fault free and faulty conditions sufficiently similar to the actual component (system) to be useful in fault studies? 4) Can the model be incorporated into a digital computer simulation involving the interconnection of models of different physical components?

Summarily, a prevailing theme is the specification and development of a model or models which can be used to construct networks on which analytical and computer oriented fault studies can be performed. The purpose of these studies is to predict both fault free and faulty performance of hardware networks in the equipment. This performance information is used to develop and assess the effectiveness of tests which will ultimately be used on the real system.

The third major objective mentioned at the beginning of this section is the development of computer oriented diagnostic methods for three classes of systems. Initially, it was envisaged that a single method would evolve and that this method would be applied in toto to the system diagnosis problem. What has eventuated is a series of individual methods which are related by three features: 1) All of the diagnostic methods depend on some form of parameter variation method to simulate faulty behaviour of time domain systems. 2) All diagnostic information can be obtained from models using algorithms which can be coded into digital computer programs and 3) All signal information that must be measured in the course of a diagnostic test is in discrete (sampled) form. In Chapter 2, we develop a new class of techniques for analogue networks based on this discrete information approach. It is also assumed that all systems are operating in the real time domain and that only those signals available within the system can be used as inputs.

Time limitations and lack of a suitable hybrid system for experimentation precluded an experimental check on the validity of various techniques. However, several computer programs were written to obtain data

and experiment with various test data development simulations. Two of these programs will be described. One, in this chapter, is a general parameter variation method which develops quantized-sampled fault detection isolation data for linear and non-linear analogue networks and for certain types of interface networks. The second program, described in Chapter 3, is one which consumed a major portion of the programming and computing time. It is used for combinational network fault simulation and test data generation.

1.4 TERMINOLOGY

Most disciplines have evolved their own language. The language is tailored for the discussion and explanation of ideas pertinent to the discipline. Because fault diagnosis has been applied to a variety of different systems, each discussion has tended to ascribe a somewhat different meaning to the same word.

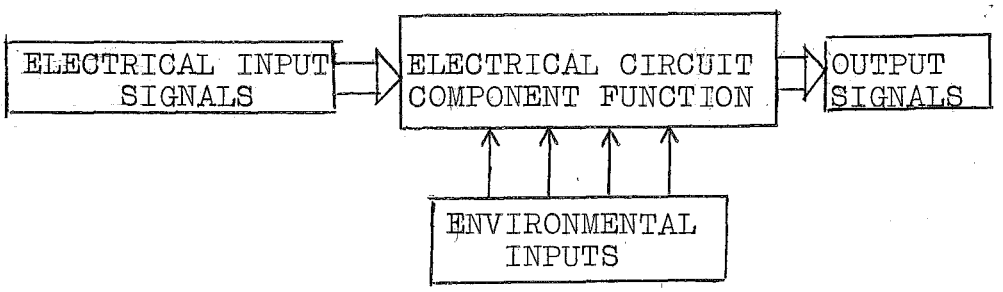
In this section, some of the important terminology and definitions are presented. The definitions have been stated with sufficient generality to apply to most of the discussion. In later chapters, we may restate, reinterpret, expand or qualify some of the terminology presented here to allow for individual system differences.

Diagnosis is the overall problem of defining a terminology, developing models, devising tests, and evaluating tests for performing fault detection and fault isolation on a system.

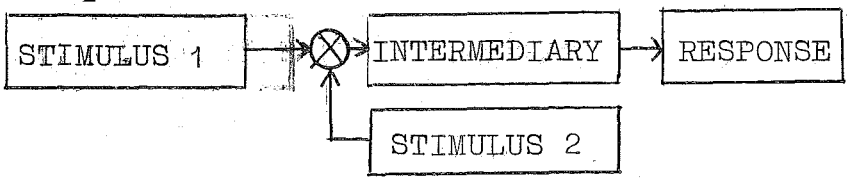
A system (see Reference 50) is any interconnection of electrical circuit components which performs a prespecified operation, or defines a mapping or function

set of signals called inputs to give a response signal or signals at the points or terminals called outputs which are a priori designated by a designer. A system has inputs or input signals which represent electrical signals which can be manipulated and outputs or output signals which represent some of the signals processed by the system. In addition, a system has intermediate signals which are not generally available for observation. There will also be inputs termed environmental inputs which, are not usually electrical, not generally measured and whose characteristics are statistically described. It is these inputs which cause temporary or permanent changes in physical parameters. Tests are the method for observing the effects of environmental inputs.

A diagnostic model is a quantitative representation of a system in the form of a mathematical model or simulation model which contains the interrelationships between input, electrical circuit components and outputs. In diagnostic models, we will assume that environmental inputs influence the functioning of the electrical circuit components and hence the outputs. The following diagram illustrates how the cause-effect relationship is supposed to exist:



The interpretation is:



The RESPONSE is observed as output from the INTERMEDIARY which is influenced by STIMULUS 1 and STIMULUS 2. Electrical signals flow horizontally, environmental signals vertically.

A network is a system whose structure can be defined by a network graph[118]. The characteristics of physical components in a network can be described electrically by lumped equivalent models. The purpose of restricting discussion to this class of systems is first to eliminate distributed parameter networks and second, to guarantee that the resulting classes of networks contain uniquely identifiable components. Because we are going to simulate parameter variations in fault studies, it is necessary to be able to identify a particular component (parameter) without ambiguity.

A variable is a quantity whose instantaneous value may be represented by a real number. To be completely general (but not completely rigorous), physical quantities such as length, time, mass, and change may be taken as physical variables. Here we are not concerned with a fundamental set of physical variables but rather with giving several examples of physical variables. The physical variables which are important in electrical system diagnosis are voltage, current and time. Parameters will be regarded as variables whose value is determined by time and environmental variables. In essence, diagnosis as it will develop, is concerned with using measurements or computations of certain variables (inputs and outputs) to indirectly estimate the value of other variables (parameters).

A signal is a voltage or current variable whose instantaneous value or magnitude comes from the set of

real numbers. A voltage signal is strictly a scalar function of time, being the potential difference between two points. A current is a vector function of time having both direction and magnitude. Because currents in networks with which we will be dealing are confined to a wire connecting two points, current is constrained to two directions. The convention will be to assume that a current signal is in the direction associated with an arrow located alongside the signal terminal or by an arrow attached directly to the signal terminal or wire. Symbols for signals will be: S, S_i, S(t), v(t), l(t), I, O, x, u y, z variously and their significance or meaning will be defined at the time of usage.

A <u>parameter</u> has very broad interpretation in this thesis. It will be defined as a variable whose value comes from the set of real numbers R^* or from some specified subset of R^*. A model parameter may correspond one-to-one with a physical parameter such as resistance or capacitance or it may be a dimensionless quantity which is used in a model to artifically represent changes in signal magnitudes or in physical parameter values. Or, a parameter can be used as an adjunct variable which operates on other parameters or signals to form a new signal or parameter value. Parameters will be written generally as p_i; other notation will be introduced as required.

A <u>physical law</u> will be defined here as the quadruple (S_1, S_2, p, M) where S_1, and S_2 are directly or indirectly measurable real valued physical variables and p is a real valued proportionality constant called the physical parameter and M is a rule involving algebraic operations which relates values from S_1, S_2 and p. Some examples are the following:

Form of physical laws $S_1 = pS_2$

Examples of some well-known (electrical) physical laws:

VOLTS = RESISTANCE . CURRENT

CHARGE = CAPACITANCE . VOLTS

VOLTS = INDUCTANCE . DERIVATIVE (CURRENT)

It will be assumed that a physical law can always be expressed in mathematical form, that it applies to an ideal physical component, and that the relationship between the signal variables is established through a proportionality constant which can be associated with a set of physical characteristics. In most cases, the parameter is not constant but will depend on environmental factors and on the signal levels etc. In other words, the above examples showing the relationship between signals and a dominant physical parameter such as resistance or capacitance are ideal. Physical elements exhibiting only one of the relationships is said to be an ideal element (device). In a physical electrical network - sometimes called an electric circuit - all of the physical voltages, currents and parameters are related. A model which attempts to include all of the operational features of a particular circuit or system is said to be an isomorph. We will assume that a model which contains all of the detail necessary to compute accurately and with certainty the total behaviour of the corresponding physical network is an isomorph.

A functional law is in a sense far more general than a physical law. We will assume that a functional law represents the cause-effect relationship between a set of voltage or current signals (inputs) and voltage or current signal called the output. A functional law

can be represented by a single mathematical operation, operating on the inputs or by a composite (and probably complex) mathematical operation on two or more variables or functions. The complexity of the operation(s) is "masked" for discussion purposes by saying that the output is a function of the inputs. The reader may now wonder about the use of law in this context; a functional law is really an artificial term invented to describe the processing activities of such electrical devices as summers, amplifiers, logic gates and more complex devices such as integrated circuit logic networks or sub-systems. When a grouping of physical elements is brought together to form a many-one transformation of the variables, most of the individual variables are disregarded and only the gross input-output properties are of interest. Detail is sacrificed to obtain a simplification. A model which has been reduced from an isomorph to the gross functional form will be called a homomorph.

1.5 SYSTEM DIAGNOSIS: MODEL, SOLUTION, AND INTERPRETATION

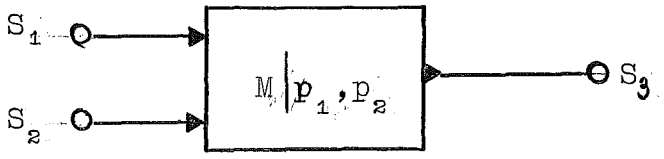
The purpose of this section is to illustrate some of the salient features of developing and interpreting diagnostic information; we use an example.

Although not completely comprehensive, it illustrates several of the problems of fault detection and isolation in systems. The system in this example is a simple functional element model consisting of two input signal lines to which voltage signals may be applied. It will be assumed that the values of the signals come from a finite set of real numbers. The output terminal has an associated voltage signal whose value depends on the input signal values, on the system's internal component

values, and on their interconnection.

If an isomorphic model of the system is available, a detailed electrical analysis of the system using conventional network analysis techniques can be made. In this example, an homomorphic transformation will be imposed which gives the output signal values in terms of a mapping, M. The electrical details of the behaviour are obscured. "Internal component conditions" are represented by parameter value settings.

The system, which is shown in Figure 1.3, has two input signals S_1 and S_2 whose values are from the sets $S_1 = \{0,1,2\}$ and $S_2 = \{1,5\}$. The output signal S_3 has values given by $S_3 = \{0,1,5,10\}$. Associated with the system are two parameters p_1 and p_2 whose values will be taken to be from the set $\{1,2\}$. The operation of the system is memoryless and its output is determined by a mapping represented by M.



Memoryless Functional Element Model
Figure 1.3

S_1 and S_2 are shown with arrows going into the system S_3 is shown with an arrow pointing out. The convention will be to show all throughput signals (regarded as independent) pointing into the block representing the system. Output signals (regarded as dependent) may have arrows pointing out of the block. If the direction of signal flow is unambiguous, the arrows will be dropped. The operation (mapping, function, relation) will be written inside the block. In Figure 1.3, the notation

$M|_{p_1 p_2}$ denotes the fact that a mapping M on (S_1, S_2) by the system depends in some particular way on two parameters p_1 and p_2. For this example, we assume that the domains of S_1 and S_2 are different. These ideas can now be put on a more concrete basis. First we enumerate some of the properties of set composition that will be used here to construct the "operation" of the example system. ($\times$ is the Cartesian product)

(a) The 'input set domain written $S_1 \times S_2$ is all ordered pairs of signals S_1 (first) and S_2 (second):
$S_1 \times S_2 = \{(0,1),(0,5),(1,1),(1,5),(2,1),(2,5)\}$.

(b) The output signal domain for S_3 is given:
$\{0,1,5,10\}$.

(c) The parameter values come from the set which we introduce here as the product set $p_1 \times p_2$
$= \{(1,1),(1,2),(2,1),(2,2)\} = \{\gamma,\alpha,\beta,\delta\} = \Pi$.

A conventional method for explicitly representing the various mappings performed by the system is used in Figure 1.4. We imagine that the four mappings illustrated correspond one-to-one with the actual system under the conditions that both the model and system input signals come from the product set $S_1 \ S_2$, and that the internal system conditions are modified to correspond to model conditions implied by the parameter settings given by $\gamma,\alpha,\beta,\delta$. Then for all inputs and parameter conditions, if the model outputs are identical to the system, the correspondence is complete. As noted previously, this model is an example of an homomorphic functional element.

Figure 1.4(a) shows the notation which is commonly used to depict a mapping. A definition may be helpful

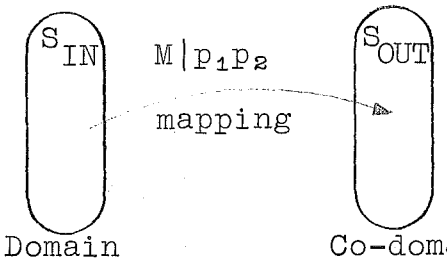
at this point: We define a mapping as the assignment of elements in one set called the domain to elements in a second set called the codomain. Using the general input and output sets S_{IN} and S_{OUT}, the mapping shown in Figure 1.4(a) can be written as $M|_{p_1,p_2}: S_{IN} \rightarrow S_{OUT}$, or $S_{IN} \xrightarrow{M|_{p_1,p_2}} S_{OUT}$.

The four mappings shown in Figure 1.4 (b), (c), (d) and (e) are all different. They may be thought of as mappings which model the performance of four different systems of the type shown in Figure 1.4. Alternatively, it is suggested that the figure can be interpreted as four mappings which are conditioned (parameterized) by the system parameters. One mapping $M|\gamma$ ($\gamma \Rightarrow$ good) can be thought of as the <u>good</u> system mapping and the others as <u>bad</u> (faulty) system mappings. If it is known that the assumed parameters are constrained to be the values given by the four pairs, Figure 1.4 may be regarded as a complete model of the system. Both good and bad versions are shown.

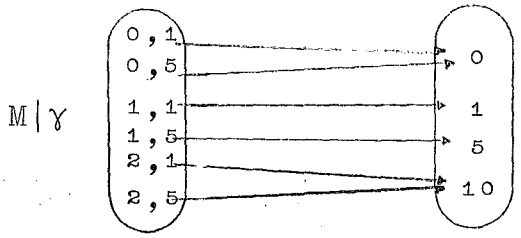
The reader may observe that the good system behaves very much as a multiplier for real numbers. The exception is the input element (2,1). Because the co-domain is missing the element 2, the system is not a multiplier. We reiterate several points and make additional important observations about the mappings in Figure 1.4, which are:

1) The values of the parameters p_1 and p_2 condition the mapping M.

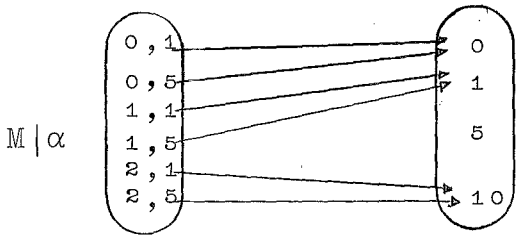
2) It is not generally sufficient to look at only one input-output response to know if the mapping is conditioned by γ, α, β or δ. For example, the input (1,1) always causes the output to be 1, independent of the particular mapping.



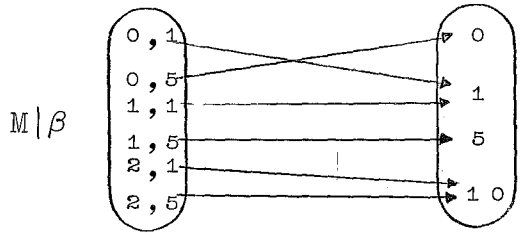
a) General graphic mapping notation



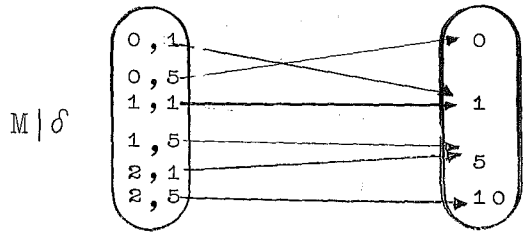
b) Mapping with parameters set to (1,1)



c) Mapping with parameters set to (1,2)



d) Mapping with parameters set to (2,1)



e) Mapping with parameters set to (2,2)

Functional Element Mapping
Figure 1.4

3) The mappings performed by the system are not one-one. That is, the element in the codomain can be the image of more than one element in the domain. For instance, 10 is the image of both (2,1) and (2,5) in three of the cases.

4) Not all of the mappings are onto.[82] In case (c) 5 is the image of no element in the domain.

This example has served to illustrate the following points about a network model:

1) The output depends not only on the input but also on the parameter values which are affixed to the model in a way which alters the output solution, depending on their values.

2) Input-output mappings may not be unique for the good model or for its various bad versions. Consequently, a single solution may be insufficient to decide which parameter settings should be associated with the output solution.

The example does not show how the effect of computational noise might be treated nor does it demonstrate the relationship of the model to its physical equipment counterpart. In the following section, diagnostic models and methods for developing models are discussed which yield solutions that are computationally equivalent to the equipment. Section 1.7 discusses tests in relation to the equipment and the diagnostic model of the equipment. Questions of computational accuracy are discussed in that section.

1.6 MODELLING: REPRESENTATION AND SIMULATION OF FAULTY EQUIPMENT

A model is a representation of a physical element, component, device, equipment or phenomenon that describes the relationship in space and/or time between two or more of the physical variables. In this section, we describe two primary electrical element model classes. These models are generally classified as isomorphs or homomorphs. Interconnections of these primary diagnostic models specify a particular equipment network model configuration whose performance both for fault free and faulty conditions can be computed.

Models presented here and applied in Chapters 2, 3 and 4 are mathematical expressions relating dependent output signal values to independent input and parameter values. Mathematical expressions are in algebraic or simulation form. Faults may be simulated in the model by prescribing parameter settings whose values correspond to a certain fault condition. The fault conditions modelled (perhaps we should say simulated) are those designated by the fault policy.

The procedure for developing complex models from simpler models is important. We begin by showing how detailed isomorphic primary models can be reduced to slightly less detailed homomorphic models. This reduction enables us to develop a hierarchy of equipment component models. For example, the component (we use the word element also) might be a transistor, an integrated circuit amplifier or an even larger interconnection of these elements. Then, because the fault simulation policy has a direct effect on the <u>type</u> and level of the diagnostic model, we illuminate some of

the conditions which restrict our ability to determine internal network parameter values by looking at a few of the external outputs. Important here is the notion of an output observable fault. In Section 1.6.3 a definition of the equipment partitions which will be used for diagnostic studies is presented. Because the diagnostic model is generally more detailed than the performance model needs to be, and because the parameter variation effect is to be preserved, a description of the useful primary models and their application to diagnostic modelling is discussed in some detail. The reader is advised that although this discussion covers several pages, it is in a sense incomplete. All possible models of network components cannot be covered and indeed, it would not be useful to try to do so in a discussion such as this. Rather, the ideas of modelling and reduction and the representation of equipment using these models which is presented in this section are those which are relevant to diagnostic techniques. Each technique, whether it be data or test development will require a particular set of models.

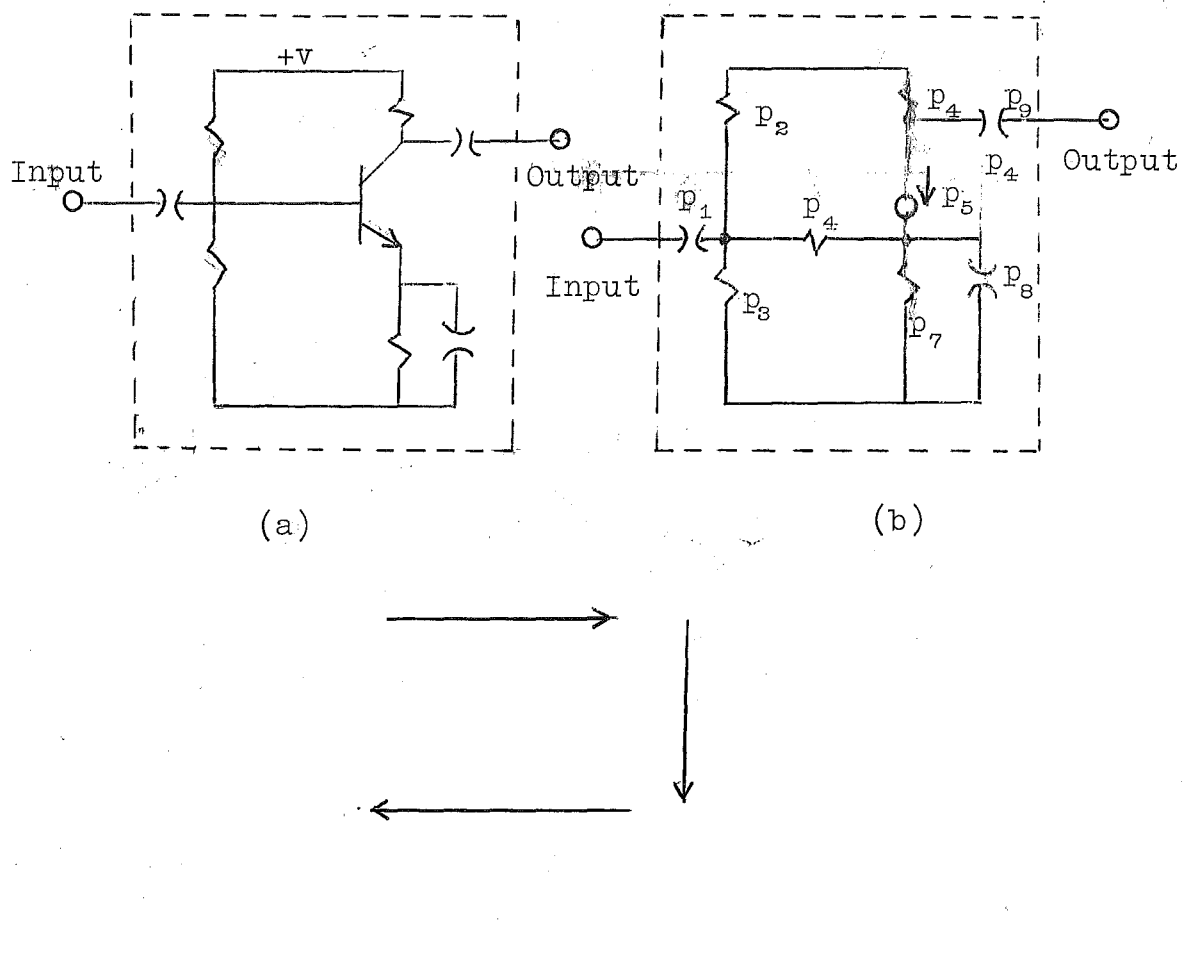
1.6.2 Transforming Isomorphs to Homomorphs

In Chapter 0, Section 0.4, diagnostic models were introduced. The significant properties of these models were stated and a model hierarchy was presented. The importance to diagnostic studies of developing a model which accurately mirrors the fault characteristics of physical components was pointed out. Because model formulation is partly science but mainly art, the overall model development must be evolutionary; the system is partitioned into sub-systems of proportions consistent with test objectives, a fault policy is

stipulated, a diagnostic model for each sub-system (or its constituent networks and their sub-networks etc.) is formulated and finally an evaluation of the models is carried out. Changes are made until a satisfactory set of models is produced.

Diagnostic model development must rely on judgment. If the replaceable unit in the system is the individual resistors, capacitors, inductors and transistors, then the diagnostic model should contain parameter information on these basic electrical network components. We shall refer to these as components or as primitive (electrical) network elements. The word parameter refers either implicitly or explicitly to the value of resistance, capacitance etc. assigned to the primitive element. An objective in diagnostic modelling is to develop a model whose parameters correspond in a known way to the physical component parameters. This objective is achieved for the simple elements but becomes increasingly difficult to attain as the complexity of the model increases. This complexity can be effectively reduced by using a simplification which we now describe.

Figure 1.5(a) shows a simple single stage transistor amplifier schematic and (b), (c) and (d) some of the possible models for representing its operation in terms of input output signals. The detailed model in Figure 1.5(b) contains parameters p_1 though p_9 assigned to a particular model configuration of the network. This equivalent circuit model is the graphic representation of the mathematical model. A particular amplifier is modelled by assigning values to the parameter. If desired, many different amplifier solutions can be obtained using parameter values selected from a statistical frequency distribution of the possible parameter



Reduction Procedure

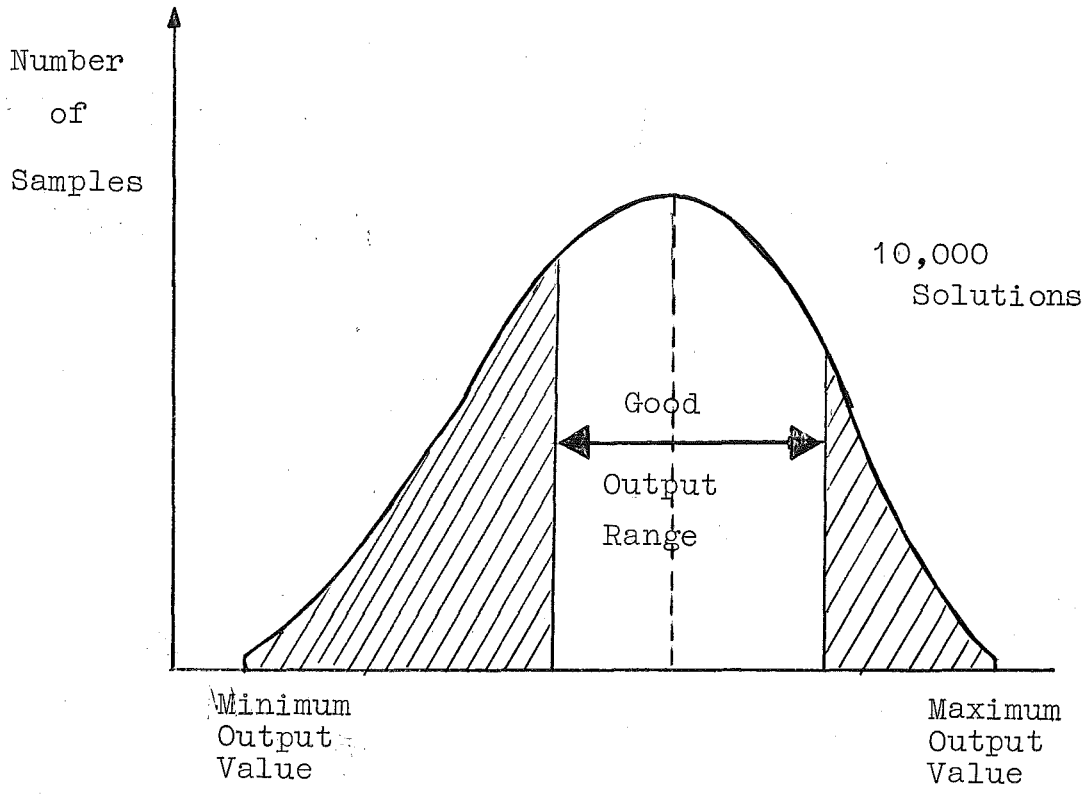
Figure 1.5

values. Solving the network many times over using the so-called Monte-Carlo simulation results in an output value frequency distribution like the one shown in Figure 1.6. It is assumed here that parameter value distributions are known and that the input signal frequency is fixed. The cross hatched areas in Figure 1.6 indicates the solution values outside the good output range. In other words, the combinations of parameter values which lead to solutions within the cross hatched area correspond to fault conditions. Ten thousand different solutions might be required to obtain the distribution shown in Figure 1.6. To record the values of the parameters which give output values outside the good range would require excessive computing storage. Because of the possible infinite number of combinations of parameter values which give output solutions outside the good range, for test information development it is necessary to limit this number. This is done by making the assumption that the parameter values change one-at-a-time.

The number of model parameter value combinations required to develop diagnostic information can be decreased in another way. In Figure 1.5 (c), a black box model with only four parameters h_1, h_2, h_3 and h_4 is postulated. This model can be developed from Figure 1.5 (b) by computing the two-port equivalent of the network. The objective of this model transformation will be to reduce the number of solutions required in the diagnostic test and data development area (see Figure 1.0) and to limit the number of parameters which are required for a particular diagnostic model. The total effect of assuming one-at-a-time variation and model simplification is to reduce the number of solutions which must be computed while at the same time obtaining a model which

is input-output equivalent to the fine model. This is, of course, necessary and desirable from the computer storage media standpoint. In most situations, it is possible to accurately compute the ranges of the h_i from knowledge of the p_i or perhaps predict the likely variations in h_1, h_2, h_3 and h_4 without the detailed analysis. In other cases, it may be necessary to estimate the values because the complexity of a more detailed model hinders accurate analysis or for reasons of engineering expediency. It is, of course, necessary to be able to evaluate the model selected to ascertain that it conforms input-output wise as predicted. Evaluation is a time consuming and sometimes difficult task. In some instances, an even more coarse model such as shown in Figure 1.5(d) may be sufficient. In this model, there is a relationship to the detailed primitive electrical network elements p_1 through p_9. But because the model is twice reduced, the relationship may be remote. However, if one-at-a-time parameter changes are assumed, for p_1 through p_9, it is possible to compute a value for h_1, h_2, h_3 and h_4 and subsequently for G. (G, may in fact, be just one of the h_i.) In each model a change in parameter value is due to a change in some physical feature in the network at either the microscopic level or the macroscopic level or both. To generalise, as the model description gets finer, the relationship between a model parameter and a physical network parameter becomes progressively clearer.

In this investigation the clearest picture we obtain is the electrical circuit model representing resistors as resistance, capacitors as capacitance, inductors as inductance, sources (controlled and independent) as ideal sources and resitance, capacitance and inductance. This model has previously been termed an isomorph. Devices



Monte Carlo Solutions Showing Tolerance Spread

Figure 1.6

such as transistors and diodes will themselves be coarsely modelled as interconnections of these primitive electrical network elements. If the parameters in the network model are one-to-one with the physical network components, accurate fault analysis at the primitive network element level is possible. For integrated circuits, the network model development is complicated by the distributed nature of the physical network. Nevertheless, models can be produced which accurately represent input output performance of complex integrated circuit structures. We always assume that an accurate lumped parameter model can be formulated. However, because the integrated circuit is probably the throw-away package, an homomorphic diagnostic model is sometimes justified. Such a justification must be founded on the type of integrated circuit, and the ultimate application of the circuit and the model.

1.6.1 Effect of Fault Policy on Model Development

The fault policy is a key factor in determining the form of the diagnostic model. If the model must simulate a variety of fault conditions and if the effect of the fault on the output is marginal or difficult to predict, an isomorphic model will be required. Again, if the faults have a predictable effect on the output and if the effect is manifested simply, then the model for this situation can be correspondingly simplified.

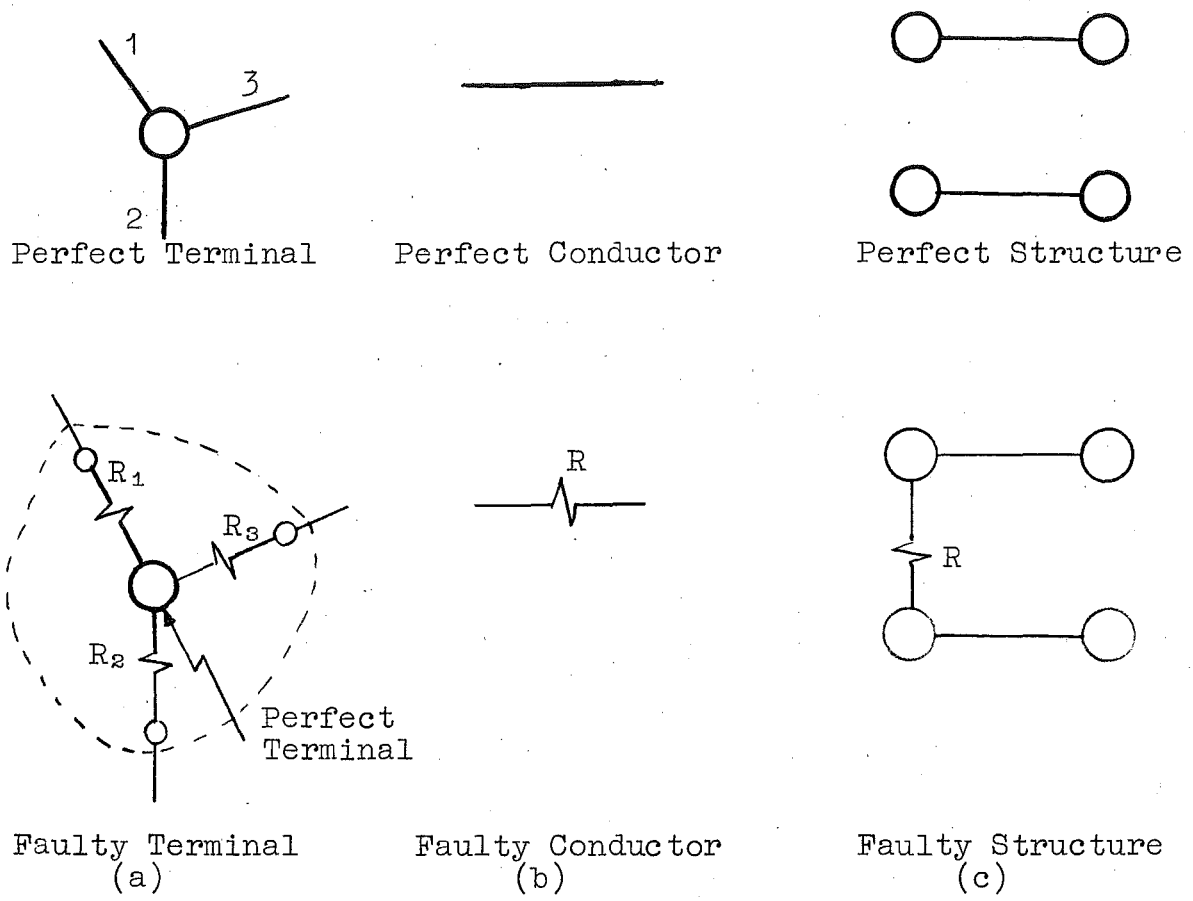
In hybrid systems discussed here, two types of faults are assumed to prevail; these are the catastrophic and the degradation. In analogue networks, a catastrophic fault can be regarded as a case of extreme degradation. Models which can simulate degradation faults can usually be used to simulate catastrophic faults. Digital

networks are non-linear and tend to be relatively insensitive to degradation faults. Consequently, they can be modelled by elements with catastrophic failure simulation modes only.

Perhaps the most difficult decision to make in specifying a fault policy is that of deciding how likely certain faults are and where faults will occur. The likeliness of a fault, its time of occurence, and its rate of change once it starts are usually assumed random. An expedient and usually reasonable assumption is that all faults occur one-at-a-time. This <u>one</u> fault assumption is desirable and is valid for digital networks [7,40]. It is less reasonable for analogue networks but leads to tractable methods which produce data which is useful in many cases. Actually, the one fault assumption is not really a limitation of the model. A detailed model has the capability of simulating almost any combination of fault conditions. This assumption is invoked to limit the number of solutions hence, quantity of data required for TDD.

The question of where faults will occur is as difficult to answer as when we have restricted the sources of faults in discussions in Chapters 2,3 and 4 to the component component body. In other words, terminals and conductors are assumed perfect.

It was mentioned in Chapter 0 that terminal, interconnecting conductor, or structural change faults can be easily treated by modelling. We will not deal specifically with them here other than to note how they might possibly be modelled. Figure 1.7 illustrates the method. In Figure 1.7 (a) a perfect terminal with three outgoing conductors labelled 1,2 and 3 is shown in the



Perfect Terminal | Perfect Conductor | Perfect Structure

Faulty Terminal (a) | Faulty Conductor (b) | Faulty Structure (c)

Fault Representations for Terminals, Conductors and Structure
Figure 1.7

top illustration. In the bottom illustration, a model for representing the faulty terminal is shown. Resistance values R_1, R_2 and R_3 are inserted. Under good conditions $R_1 = R_2 = R_3 = 0$. A fault might lead to a model condition $R_1 = R_2 = 0$, $R_3 = \infty$. This set of values would simulate an open circuit (e.g. cold solder joint or cracked terminal) condition for conductor 3. Intermediate values of R_3 would indicate a degraded condition. The interpretation for Figure 1.7 (b) is immediate. The perfect conductor has $R = 0$. A degraded conductor corresponds to $0 < R < \infty$, and an open circuit condition is $R = \infty$. These values are of course ideal; realistic values can be used if they are available. Figure 1.7(c) shows a representation for one structural fault. The faulty structure is equivalent to the perfect structure when $R = \infty$. The structural fault might be due to a lessening in circuitboard resistance, a decrease in subsstrata resistance etc. It is easy to imagine the complexity that can be built up from a fairly simple circuit model if terminal, conductor and structural faults are simulated.

1.6.2 When is a Failure a Fault?

In the elementary example illustrated in Figures 1.3 and 1.4, it was shown that a fault in a network component may affect the network output value(s) for only some of the inputs. There are two important interpretations of this condition. One interpretation is a condition where output signal values are identical for <u>some</u> of the input signal combinations, independent of whether the network is good or contains a certain fault. For example, in Figure 1.4, inputs (0,1) and (0,5) give output 0 both for the good (γ) network and for the fault α; other cases of identical input-output pairs for different $\{p_1 p_2\}$ can

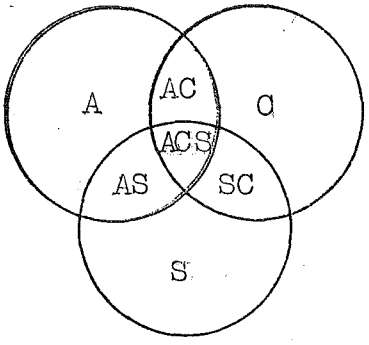
be observed. This type of "not always output observable" fault is typically found in combinational and sequential networks. The second class of "not output observable" faults can exist in frequency dependent networks whose input signal frequency is too low or too high to excite a particular mode. In this case a component parameter value change which is dominant in determining only one of the modes may not be sensed at the output if the input signal frequency is too high or too low.

Another possibility which can occur in analogue and interface networks should be mentioned. Because equipment is designed with signal value tolerance limits specified, a drift in the parameter value of one of the network components may not cause any output values outside the tolerance limits. Consequently, a parameter value outside its specified tolerance limits will not always cause the output signal(s) to deviate from allowable limits. The worst case design philosophy is often applied to design networks which can tolerate component parameter value drift[63].

1.6.3 Diagnostic Modelling Procedures

To develop a regime of diagnostic models for a particular equipment, data on likely component failure rates, modes of equipment failure, parameter value drift and environmental conditions must be compiled. This information, used in prescribing a meaningful fault simulation policy, also shapes the diagnostic model. It is, however, not the only important information. The diagnostic model builder must be cognizant of such factors as equipment complexity, ultimate operating environment of the equipment, and costs associated with

developing diagnostic information and applying it to equipment checkout. These factors influence the decision which specifies at what level(s) fault detection and/or isolation within the system should be. The level stipulated, be it major sub-system, sub-system network, network functional element or functional element component or a combination will lead to a requirement for different models. Subsequently, the model or models suitable for the level(s) must be developed. This level selection can best be done by partitioning the equipment into units as shown schematically in Figure 1.8 and by subsequently developing models for the units.



Legend

A - Analogue Equipment

C - Combinational Equipment

S - Sequential Equipment

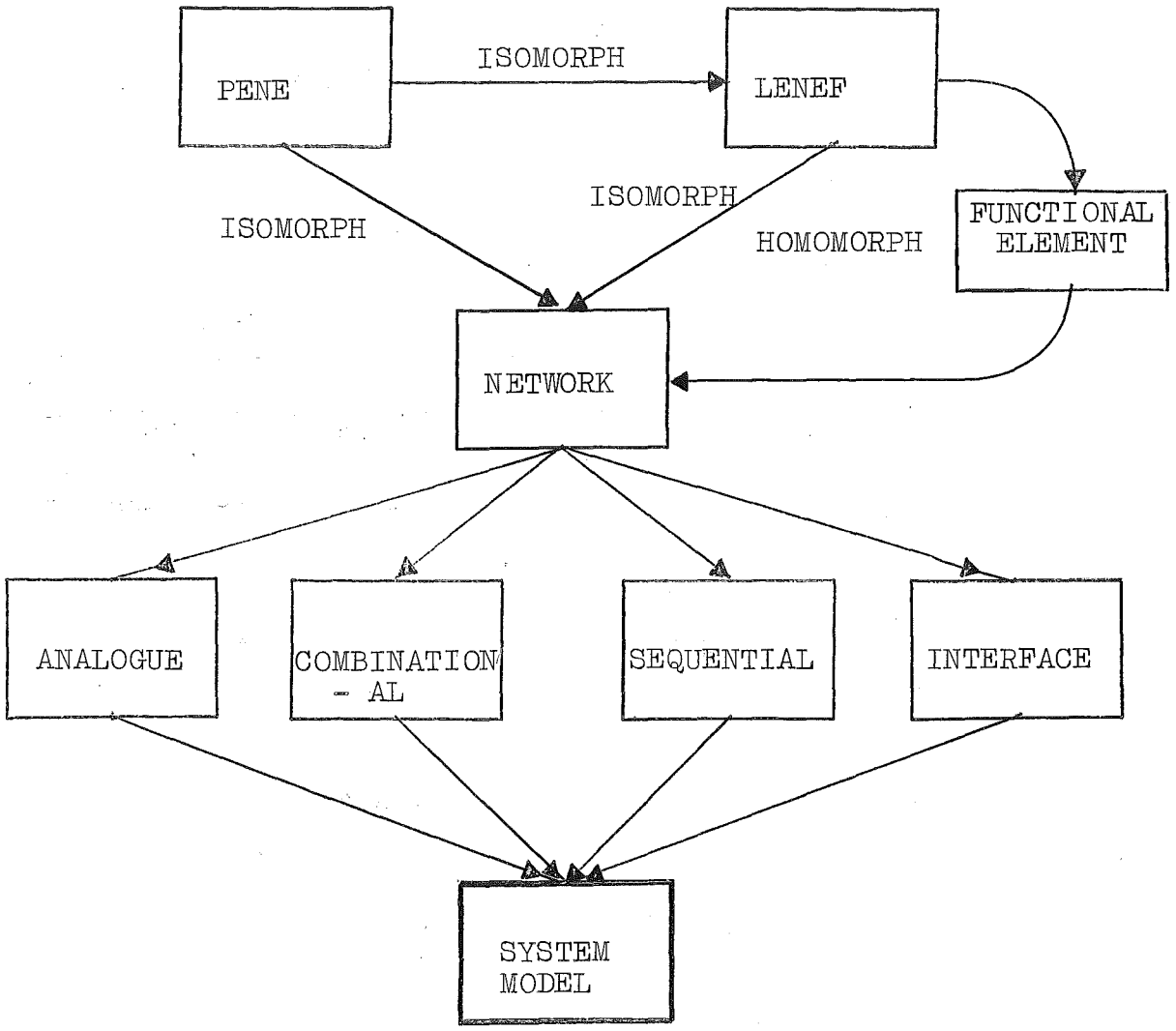
Interface equipment shown as a combination: AS, AC, SC, or ACS

Schematic of Equipment Partitioning

Figure 1.8

We will imagine that equipment levels or partitions and their corresponding models can be identified.

Figure 1.9 presents a breakdown showing relationships between levels of diagnostic models. Models correspond to equipment partitions. The most complex is the system model, the least complex is the primitive electrical network element (PENE). The top of the diagram in Figure 1.9 shows how PENE can be used to synthesize models of linear and non-linear electrical functional elements (LENEF) or to model the network



Diagnostic Model Diagram
Demonstrating Hierachical Relationship

Figure 1.9

directly. The resulting LENEF and NETWORK are isomorphic models. The detailed electrical relationship of the LENEF can be reduced as shown in the top right by transforming the LENEF into a homomorphic functional element (FE) by reducing the specific relationships between the LENEF outputs and the parameter values of the PENE of which it is composed. The FE will usually have fewer parameters. The less complex FE can subsequently be incorporated into the network which will be a homomorphic model in this case. Networks of the analogue, combinational, sequential and interface variety are interconnected (if required) to form the system model.

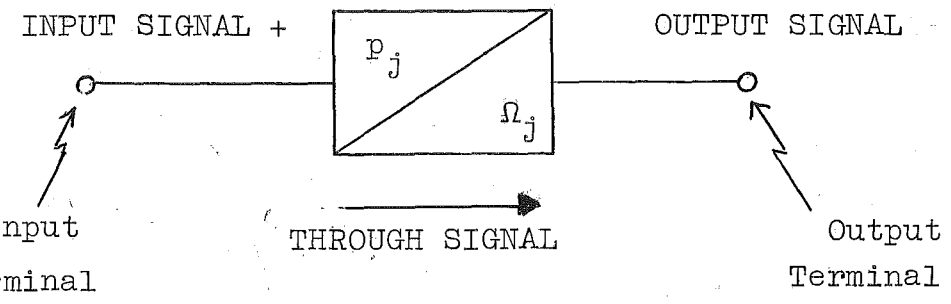
1.6.3.0 PENE and LENEF Modelling

A variety of system component models and model graphics exist[47, 145]. The block diagram[148], the signal flow graph[29] and equivalent circuit diagrams[118] being widely used in representing both structural and functional features of a system.

We require that the diagnostic models be functional representations of the equipment in the sense that they provide an unambiguous description of the throughput (voltage or current) signal transformations at all computable terminals in the model. These terminals should correspond to measurable terminals in the equipment. In other words, for given inputs the computed model output signals must be identical to those that the equipment would generate when stimulated with the corresponding physical input signals.

1.6.3.0.0 PENE

Figure 1.10 shows the general form of a basic primitive electrical network element. It consists of



Basic PENE Model
Figure 1.10

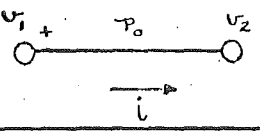
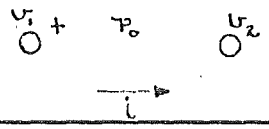
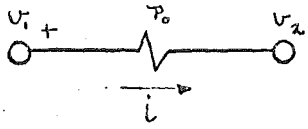
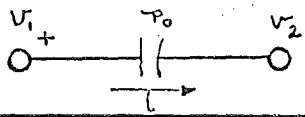
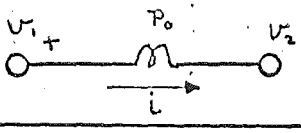
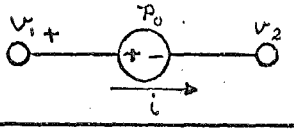
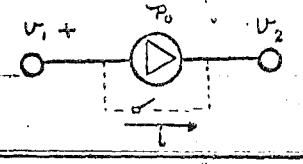
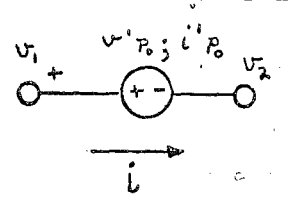
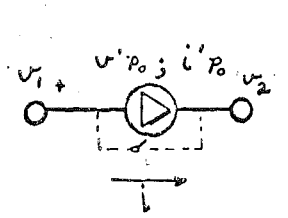
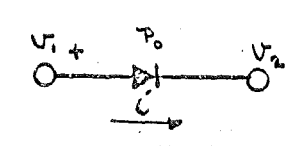
an input terminal, an input signal, an output terminal, an output signal, a through signal, a parameter p_j and an operation Ω_j for "combining" signals and the parameter. The input and output signals are potentials, which appear on the input and output terminals. The through signal is current. The terminals and conductors leading into and out of the block in Figure 1.10 are assumed perfect. The arrow shows the assumed direction for current and the plus symbol indicates that the (input) terminal is assumed to be at a higher potential than the output terminal.

A parameter*, affixed to the PENE and written p_j, operates on a signal or a combination of two of the signals. The symbol Ω_j implicitly denotes the PENE

---

*On occasions, this will be interpreted as a set of parameters or a set of parameter values which will be written $\{p\}$ or $\{p_i^k\}$, i^{th} parameter at k^{th} value.

Primitive Electrical Network Elements

No.	Schematic Symbol	Name of PENE		Remarks and Interpretation
1		Terminal		None
2		Short Circuit	$v_2 = v_1$	None
3		Open Circuit	$i = 0$	None
4		Resistance	$v_2 = v_1 - p_0 i$	Output voltage minus p_0 operating on current.
5		Capacitance	$v_2 = v_1 - p_0 \int_0^t i\,dt$	Output voltage is input voltage minus p_0 operating on integral of current.
6		Inductance	$v_2 = v_1 - p_0 \frac{di}{dt}$	Output voltage is input voltage minus p_0 operating on derivative of current.
7		Voltage Source	$v_2 = v_1 - p_0$ $0 \leq i \leq \infty$	Output voltage is input voltage minus p_0; p_0 is source value, it may be a set of parameters.
8		Current Source	$i = p_0$ $0 \leq v_2 \leq \infty$	Dotted lines indicate short circuit around current source when not "in circuit". p_0 is value of current generated by current source.
9		Controlled Voltage Source a) Voltage Controlled b) Current Controlled	$v_2 = v_1 - v' p_0$ $v_2 = v_1 - i' p_0$	v' and i' are values from another PENE.
0		Controlled Current Source a) Voltage Controlled b) Current Controlled	$v_2 = v_1 - v' p_0$ $v_2 = v_1 - i p_0$	v' and i' are values from another PENE.
1		Ideal Diode Figure 1.11	$v_2 = p_0 v_1$ $v_2 = v_1; i \geq 0$ $v_2 > v_1; i \leq 0$	$v_2 = v_1 - p_0 i \Rightarrow p_0 \to 0$ for $v_1 \geq v_2$ $v_2 = ? \Rightarrow p_0 \to \infty$ for $v_1 < v_2$

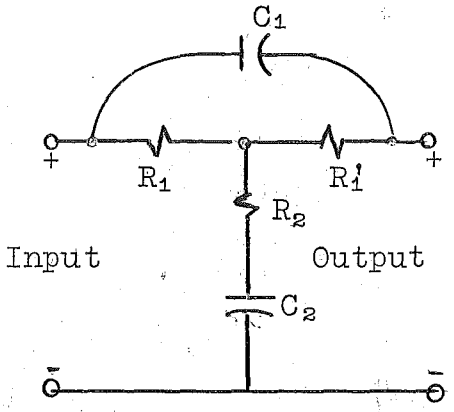
operation or mapping, the nature of which is defined by the physical law associated with the j^{th} element.

The value of any signal can be determined knowing the explicit form of Ω_j, the value of p_j and the values of two of the signals if the PENE is memoryless. If it possesses memory, then the signal value will depend on Ω_j, p_i and present and past values of the other two signals.

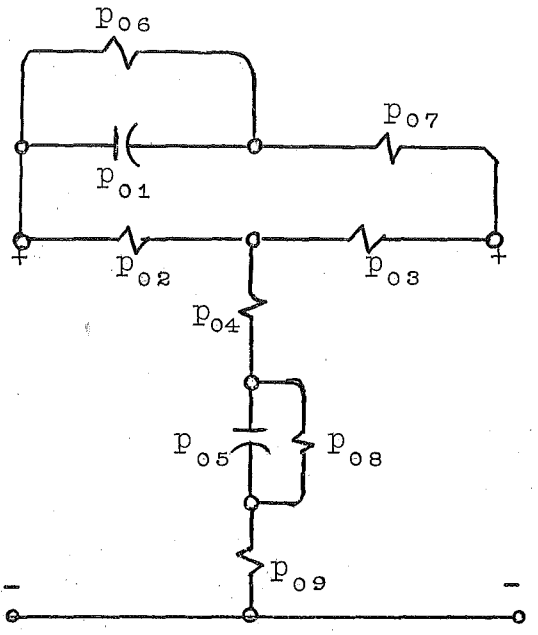
Interconnections of PENE can be made only at terminals. The resulting interconnection, termed a complex, may form a LENEF, an FE or a network. The distinction between these is sometimes simply one of definition. A complex is a network if there is at least one closed path.

The operation or mapping Ω_j performed by the PENE is assumed to be perfect. To represent a variation in the characteristics of the PENE means to change the value of p_j. Consequently, the fault policy must be stipulated to determine permissible values for p_j.

Figure 1.11 lists important PENE showing the schematic symbol, name and law governing the relation between signals and parameters. All signal and parameter values come from the set of real numbers, R^*. The relationship between the parameter value p_0 and a malfunction in the PENE requires special interpretation depending on the fault simulation policy adopted. It is important to emphasize that parameter values do not depend on signal values. They are selectable and may be set to any value, $0 \leq p_0 \leq \infty$. In some cases, a combination of degradation and/or catastrophic changes in the component parameter value are to be simulated. In many fault studies, it is commonly assumed that a



Normal Model (a)



PENE Diagnostic Model (b)

	Condition Simulated	Parameter Value Settings to Realize Conditions								
		p_{01}	p_{02}	p_{03}	p_{04}	p_{05}	p_{06}	p_{07}	p_{08}	p_{09}
1	Normal	C_1	R_1	R_1'	R_2	C_2	∞	0	∞	0
2	Degradation C_1	VAR	R_1	R_1'	R_2	C_2	∞	0	∞	0
3	Open circuit C_1	C_1	R_1	R_1'	R_2	C_2	∞	∞	∞	0
4	Short circuit C_1	C_1	R_1	R_1'	R_2	C_2	0	0	∞	0
5	Full R_1	C_1	$0 \leq p_{02} \leq \infty$	R_1'	R_2	C_2	∞	0	∞	0
6	Full R_1'	C_1	R_1	$0 \leq p_{03} \leq \infty$	R_2	C_2	∞	0	∞	0
7	Full R_2	C_1	R_1	R_1'	$0 \leq p_{04} \leq \infty$	C_2	∞	0	∞	0
8	Degradation C_2	C_1	R_1	R_1'	R_2	VAR	∞	0	∞	0
9	Open circuit C_2	C_1	R_1	R_1'	R_2	C_2	∞	0	∞	∞
10	Short circuit C_2	C_1	R_1	R_1'	R_2	C_2	∞	0	0	0

(c)

Bridged-T, RC Network Fault Simulation

Figure 1.12

catastrophic failure will result in open circuit or short circuit behaviour by the element. In this case, it is usual to use a series arrangement of the PENE and a resistance PENE. In Figure 1.11, a short circuit and open circuit are listed as PENE. These can be considered as degenerate cases of the resistance element with $p_0 = 0 \Rightarrow$ short circuit and $p_0 = \infty \Rightarrow$ open circuit. ($\Rightarrow$ means 'implies').

For each of the PENE in Figure 1.11, the law governing the element signal processing is invariant. However, the p_0 may be set to values which simulate fault conditions stipulated by the fault simulation policy. In some cases, as we shall demonstrate in the following example, a combination of PENE may be required to simulate different faults.

Example: This example illustrates the method for developing a network model which can be used to simulate a certain set of faults. These faults are commonly observed in networks such as the bridged-T,RC network shown in Figure 1.12. The normal model is shown in Figure 1.12 (a) and the PENE diagnostic model is given in Figure 1.12 (b). It is the same as the normal model except for PENE designated p_{06},p_{07},p_{08} and p_{09}. They have been incorporated to simulate the short circuit and open circuit catastrophic faults in C_1 and C_2. The table in Figure 1.12 (c) is a listing of parameter value settings for a particular fault simulation policy. It is compiled on the assumption that both catastrophic and degradation faults are to be simulated. The first row shows the normal settings. p_{01} through p_{05} are selected as being C_1,R_1,R_1',R_2 and C_2. Some synthesis methods will specify exact values for these R's and C's but because of inherent manufacturing imperfections there will always

be a finite range of values. In high quality resistors it might be $\pm.5\%$ and in capacitors $\pm1\%$. It is usually assumed that degradation faults cause parameter values to drift outside these limits. In the case of the physical network, whether or not the input-output response function goes out of specification is more difficult to answer. Tests will give information which aid in answering this question.

Consider the requirements for simulating a capacitor parameter value degradation. We will use C_1 in Figure 1.12 (a) to illustrate the approach. The value of p_{o1} (see Figure 1.12 (b)) will be variable and all other parameter values will be normal. For a given input and the fault simulation policy decided, the output for the network representation in Figure 1.12 (b) can be computed for a range of settings of p_{o1} corresponding to the VAR degraded values of C_1. The conditions in rows 3 and 4 are used to simulate a catastrophic fault on C_1; $p_{o7} = \infty$ simulates the open circuit fault and $p_{o6} = 0$ simulates the short circuit fault, with all other p_{oj} set to the normal value.

A short circuit, degradation or open circuit fault in the resistors can be simulated by varying the resistance parameter of the PENE. For example, row 5 shows that all values for R_1 can be simulated by varying p_{o2}. All of the other postulated fault conditions can be observed as the conditions specified by the rows in the Table in Figure 1.12 (c).

This example has illustrated the modelling procedure required to develop a diagnostic network model or LENEF using PENE interconnections. One point about the PENE listed in the table in Figure 1.11 should be reiterated. It concerns the voltage and current sources. They are

shown as having an independent parameter p_o which determines the voltage. In practice, sources may have several parameters. For example a sinusoidal generator may have a parameter for magnitude, frequency and phase. In this case, p_o should be considered as a set of parameter values, one specifying the value of magnitude, one specifying the frequency and a third specifying the phase.

1.6.3.0.1 LENEF

Using PENE and time domain network analysis techniques, detailed input-output performance for a given element configuration and various fault conditions can be computed. A particular organization is called an LENEF. An LENEF or FE may have m input terminals and one observable output terminal. We use observable to stress that the remaining internal terminals are regarded as "not available for observation".

At any time t, given the values of the input signals, the output signals of a memoryless FE are determined by the transfer properties of the functional element. A FE may be a linear, non-linear, memory or memoryless complex or network composed of PENE whose output depends on the internal electrical circuiting formed by PENE and on all inputs past and present. However, from an external viewpoint, the relationship between input and output variables is all that matters.

It is well known that some of the basic building blocks in analogue computers and in digital computers perform algebraic operations or functional transformations on the inputs to give an output. Following others, we note the importance of the isomorphism between some of

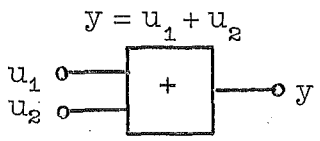
the FE interconnections and group algebra [47]. Specifically, it is desirable to retain the real number algebraic properties for linear systems. For the non-linear case, one useful set of FE corresponds to a set of electrical elements (gates) which realize a Boolean two level algebra. Two level gates which perform Boolean algebraic addition, multiplication and inversion in the set $\{0,1\}$ are important in most computing systems.

Figure 1.13 shows an incomplete but representative set of FE which can be used to model a system. Our intention is to show how these can be interpreted to simulate fault conditions. Before proceeding we list several FE and FE interconnection properties that we are assuming.

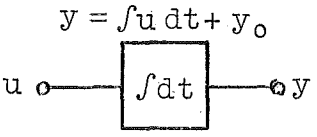
1. All systems can be modelled by series parallel interconnections of confluent (multiple input, single output) FE. In other words, a functional element can have m inputs but only one output. Consequently an n output, m input partitioning must be modelled by at least n functional elements, each of which may have up to m inputs.

2. The signals applied to a functional element must be compatible. We will arbitrarily assume that all FE have infinite input impedance and infinite output admittance. Appending PENE to the terminals will adjust this condition to give a non-ideal, realistic simulation of conditions.

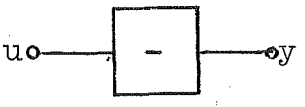
3. We again mention that all signals are treated as time dependent and belong to R^*.



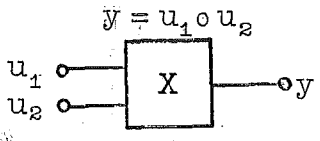
ADDITION



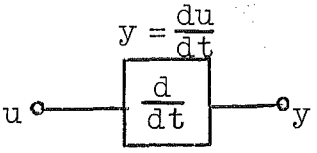
INTEGRATION



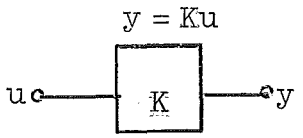
INVERSION



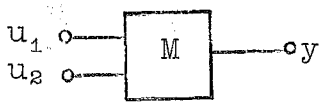
MULTIPLICATION



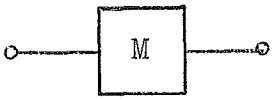
DIFFERENTIATION



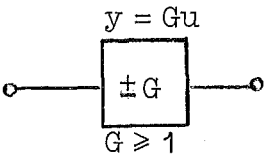
SCALAR



MAPPING



MAPPING

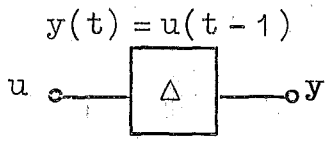


AMPLIFIER

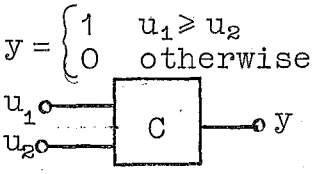
ALGEBRAIC

TRANSFORMATION

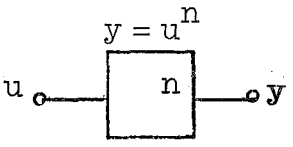
SCALING



DELAY



COMPARATOR



POWER

TIME DEPENDENT

NON-LINEAR

Basic Functional Elements

Figure 1.13

Examples : Fault Simulation Using FE

We will give two examples of fault represention in the FE. The procedure is to (a) model the FE using PENE and obtain input output behaviour, (b) reduce the number of parameters required to represent FE behaviour for both good operation and faulty operation, and (c) verify that the FE model is correct.

Example 1

A summing circuit shown in Figure 1.14 illustrates the FE, its LENEF model, and the reduced parameter FE. Starting with the ideal FE in Figure 1.14 (a), the next step is to develop the LENEF shown in Figure 1.14 (b). Note that two amplifiers of gain $-A_1$ and $-A_2$ are included in this representation. Their complexity is great but variation in their gain can be simulated as changes in the p_{oi}. Consequently, in this model, A_1 and A_2 are assumed to be constant. By using only resistive PENE in the model, we are assuming of course that u_1 and u_2 are changing slowly with respect to time. Otherwise, capacitive and inductive PENE would be required to model effects. Finally, the reduced parameter FE in Figure 1.14 (c) can be used to simulate output symptoms shown in Figure 1.15.

Output Symptom	LENEF CONDITIONS	Reduced Parameter FE CONDITIONS
Good	$p_{o1}=p_{o2}=p_{o3}; p_{o1}=p_{o5}$	$p_1=p_2=1$
$y>u_1+u_2$	p_{o1} or $p_{o2}<p_{o3}$ $p_{o4}<p_{o5}$	$p_1=p_2>1$
$y<u_1+u_2$	p_{o1} or $p_{o2}>p_{o3}$ $p_{o4}>p_{o5}$	$p_1=p_2<1$
$y=0, u_1$ or $u_2>0$	$p_{o3}=0$, $p_{o4}=\infty$ $p_{o5}=0$	$p_1=p_2=0$

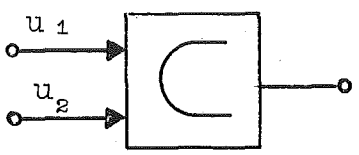
Output Symptom versus Parameter Settings for Figure 1.14
Figure 1.15

Example 2

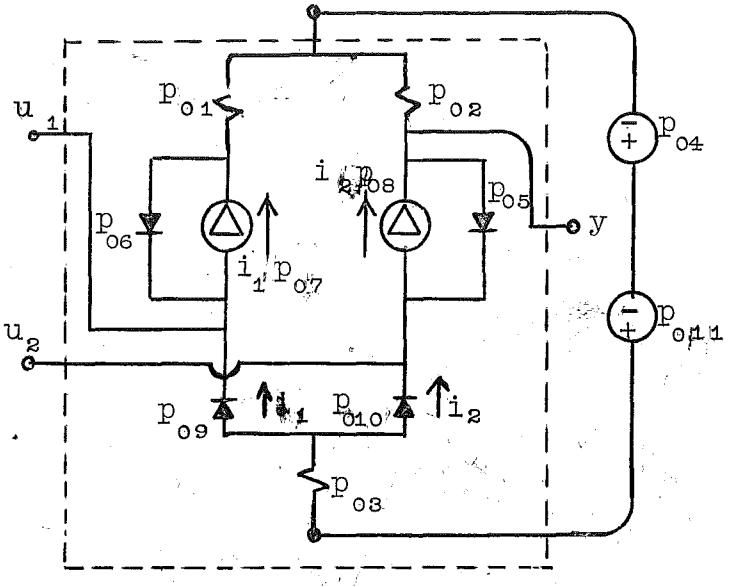
A type of comparator whose FE schematic is shown in Figure 1.16(a) has the LENEF model in Figure 1.16 (b). A detailed analysis using the fault simulation of PENE would give the behaviour of the comparator. In cases where such a lengthy analysis is not warranted, a model can be postulated having a reduced number of parameters. Using the reduced parameter FE shown in Figure 1.16 (c), different faults can be simulated. For example, the good FE is modelled (Figure 1.16 (a)) when $p_1 = p_2 = 1$ and $p_3 = 0$. A permanent output, v, is given by $p_3 = v$, $p_1 = p_2 = 0$ which is independent of the inputs. Other conditions can be simulated by selecting suitable values of p_1, p_2 and p_3.

1.6.4 Remarks on Modelling

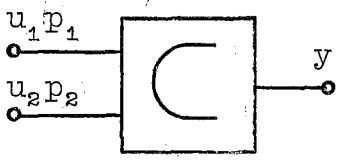
Two useful network element diagnostic models have been presented which can be interconnected to form network models. The built-in parameter variability makes these models useful for simulating fault characteristics. The networks and the models from which they are derived constitute diagnostic models. The classes of networks which are modelled for diagnostic studies will be analogue, combinational, sequential and interface. The networks are selected by an equipment partitioning which subdivides the equipment into the four main classes. For diagnostic modelling, it may be desirable to further subdivide the classes until a network of manageable proportions is obtained. A manageable network size is one for which a diagnostic model can be synthesized and solved. But consideration must simultaneously be given to factors such as fault simulation policy, objectives of diagnostic studies and whether the analysis will be done



FE
(a)



LENEF
(b)



Reduced Parameter FE
(c)

$y = u_2 p_2$ if $u_1 p_1 > u_2 p_2 + p_3$

PENE-LENEF-FE
Reduction for NonLinear Network

Figure 1.16

using hand analysis methods, computer methods or a combination. The larger the network, the greater will be the necessity to apply computer methods.

In the next section, the area of tests and test development is introduced. The treatment is general enough to be applied to different classes of system equipment. Tests that are applied to an equipment can be developed a priori using diagnostic models of the particular system class. The test development considers real system constraints on the types of faults and their relative frequency of occurrence, on availability of terminals for measurement and stimulation, and on policies concerning allowed forms of input signals. Accordingly, limitations which we impose in Chapters 2,3 and 4 on developing diagnostic information depend on the system class, inherent constraints imposed by the equipment function and configuration, and by the objectives of the test. In the following section we have endeavoured to present a general treatment of TTS, TMS and OTP which can be adapted to TDD for a given system and TOD.

1.7 TESTS

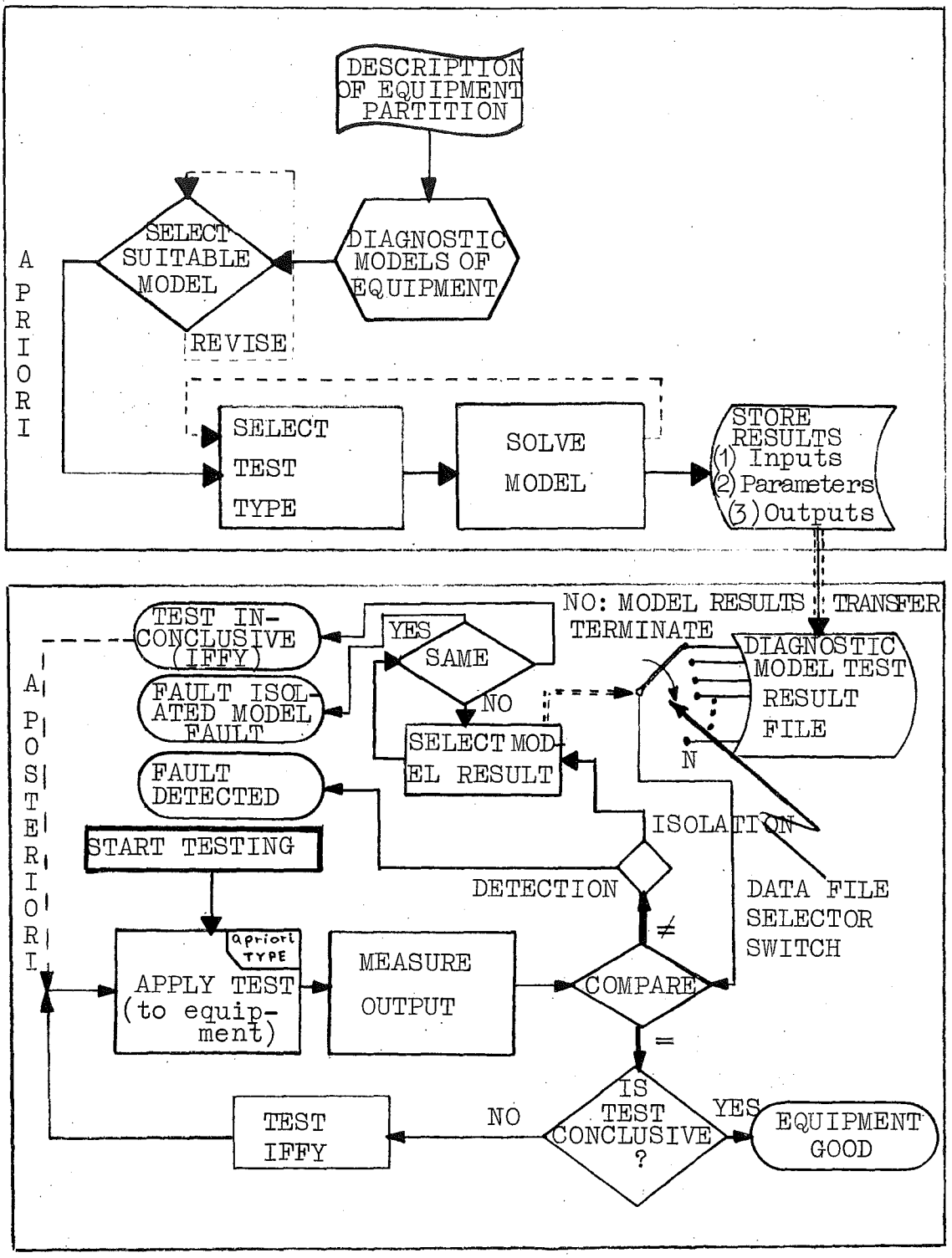
1.7.0 Introduction and Definitions

A test Θ_i is the application of an input signal, observation of the resulting output signal, and a subsequent decision. A test includes the specification of terminals, independent signals and parameter values if the test is a priori. A test a priori is the specification of an input signal function or functions for a diagnostic model, M, computation of the corresponding output signal function or sequence and a decision which classifies the model into one of several categories.

Categories to which the model may be assigned are: good possibly good, faulty or the model has parameter settings given by P^k. A test a posteriori is the application of a known input signal or signals to an equipment, E, or an equipment partition, measurement of the corresponding output signal and a decision which assigns the equipment to one of the classes: good, possibly good, faulty or fault type f_i.

A test is fundamental to diagnostic studies. It provides information for developing diagnostic testing methods and for making diagnostic test decisions. A test decision (TD) is a statement about the test data which classifies the data into one of several TD categories. Tests and testing methods are developed a priori using diagnostic models and applied a posteriori to equipment. Test data is the model input-parameter-output information and the measured equipment input-out information. It includes procedural information which is developed a priori and applied a posteriori.

Figure 1.17 shows the overall testing problem. Each block represents a diagnostic test activity or decision. The a priori TTS, TDD, TMS, and OTP functions in Figure 1.0 can be studied using the SELECT SUITABLE DIAGNOSTIC MODEL, SELECT TEST TYPE and SOLVE MODEL blocks. The STORE RESULTS block is distinguished in Figure 1.17 by a dotted connecting line labelled Transfer. This illustrates the presence of an information link between the a priori and a posteriori functions. The procedural links are not shown but will be described later in this section. The complexity of the a posteriori testing and decision making can be observed in the blocks directly beneath the a priori blocks. The reader should follow through the diagram as it will clarify the individual functions and decision blocks when they



Combining a Priori and A Posteriori Diagnosis
Figure 1.17

arise in subsequent discussion. A good starting point is the START TESTING block. We can assume that the data file selector switch is on position 1 (GOOD) to start with and that a test procedure developed a priori is applied to the equipment. The measured output for the given procedure is compared with the model output data. If the two results are equal and the test is conclusive, the equipment is judged good. If the test is inconclusive (IFFY), a second test must be applied. If a test contains a single input-output combination or sequence, the test is simple. If the test applies two or more input signal combinations or sequences, the test is termed compound. If any test gives an unequal ($\neq$) result, (see COMPARE), the top loop is activated and the FAULT DETECTION or FAULT ISOLATION procedure is invoked. The fault isolation procedure is a sequential decision process.

In this section, test type specification TTS (see Figure 1.0) is explained in relation to a priori and a posteriori test and test development. It will become clear that TDD, TMS and OTP are related to and dependent on TTS. A fundamental aspect of a test or any testing procedure is the test decision, TD. (Sometimes we will use diagnostic decision to refer to a more comprehensive decision process.) Test type specification and subsequent test decisions establish the interdependence of the a priori and a posteriori diagnosis. In so doing, they establish the cohesiveness between different aspects discussed in this work. An understanding of the interdependent can be best obtained by a) establishing an interpretation for a test which has relevance both for the model and equipment, b) by contrasting the roles of a priori and a posteriori tests in the test type specification and test decision process and c) by

specifying a test type definition which satisfies a) (above) and which can be used for analogue, combinational, sequential and interface networks. Notation will be established following the introductory remarks on test type specification.

1.7.1 Test Type Specification

The differences between an a priori and a posteriori test can be loosely described as the difference between development and application. The flexibility of the diagnostic model in simulating fault conditions is essential. Simulation makes possible a preview of the equipment fault behaviour which cannot be otherwise obtained.

It is helpful to think of a priori studies as the computation of types and locations of signal applications which produce output signal information for detecting and isolating faults. A priori studies give information that is effective and optimal. The information is effective if it will detect and/or isolate all postulated faults and is optimal if the cost is minimal. Some important costing factors are the following:

1. Time for completing a test
2. Cost of test equipment required to obtain test information
3. Confidence in test results
4. Level of skill required (manpower)
5. Sparing policy for test
6. Test-points required
7. Failures initiated by tests

Any comprehensive testability measure developed a priori must take into account the above factors.

Definition: Testability[141,147] is the ratio of failures which can be detected by a given technique to the total possible failures*. This can be written

$$\text{Testability} = \frac{\text{Detectable Failures}}{\text{Possible Failures}} .$$

Most testability definitions assume single component failures which may be either catastrophic or of the drift type. Some work has been done by Poage[115] on multiple component failures in combinational digital networks and by Seshu[127] and by Levadi[81] in analogue networks but no applications of these techniques to large systems have been reported in the literature.

The general a posteriori test type classification that will be used in this work is shown in Figure 1.18.

Test Type	Used For
Combinational (Fixed)	Fault Detection
Sequential (Serial)	Fault Detection Fault Isolation
Adaptive	Fault Detection Fault Isolation

Test Types
Figure 1.18

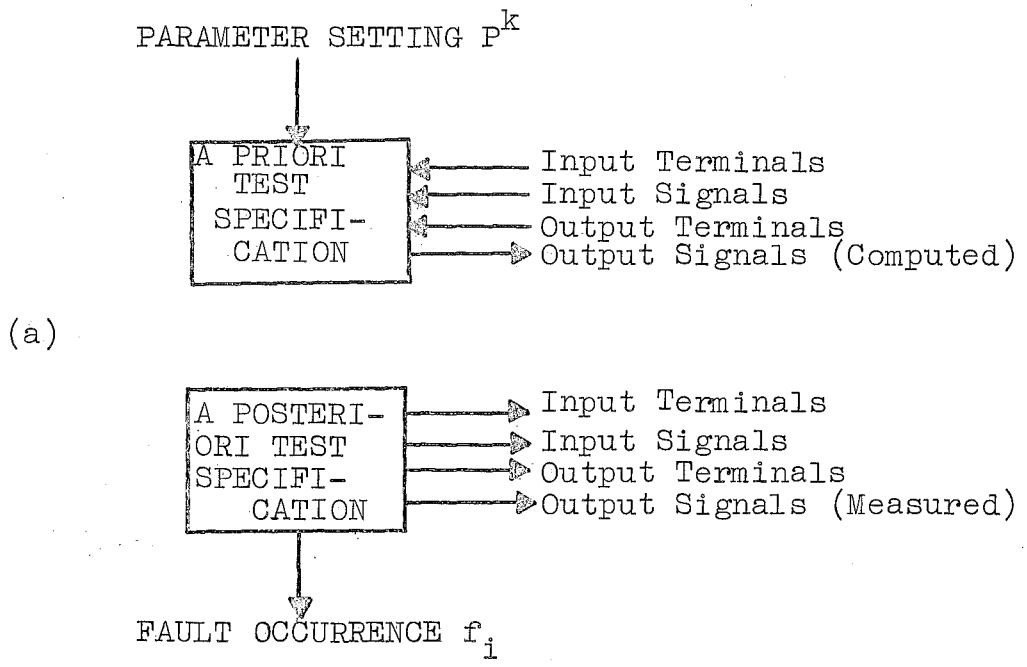
This classification does not indicate the test type selection capability inherent in a priori diagnosis. The flexibility of test specification a priori is contrasted with the "inflexibility" a posteriori in

---

*This definition can be written for isolation by substituting the word 'isolatable' (isolable) for detectable.

Figure 1.19 (a). Arrows pointing into a block indicate that this entity may be selected. These are "free-will variables". Arrows which point out of a block indicate that the quantity associated with the arrow is specified or dependent in some way on other controlled quantities. These are "non free-will variables". The primary differences between the two test type specifications have been previously stated as the difference between development and application. Other differences accentuated by the blocks in Figure 1.19 (a) are parameter setting / fault occurrence, and output signals (computed) / output signals (measured). Corresponding quantities are listed in the table in Figure 1.19 (b). Some notation is introduced now which will be applied in subsequent discussion.

We denote the vector of input signals by $\underline{u}$, the vector of output signals by $\underline{y}$ and subscripts on u and y, u_i and y_j will represent the vector components. Input and output signals are assumed to be continuous functions of continuous or discrete time and perhaps additional variables or parameters. Values of the input will be superscripted; hence $u_i^k \in U$ is the kth value of the ith component of the input vector $\underline{u}$. U is the domain of input signal values. u_i^k may be the value of the kth element of a sequence or it may simply be the value of $u_i(t)$, $t = a$ where $a \in R^*$. The output signal vector function written $\underline{y}(*)$ is used to denote the dependence of $\underline{y}$ on one or more arguments which may themselves be functions of other variables. The value of the jth output will be written $y_j^k \in Y$ where here again y_j^k may be the value of a the kth element of a sequence or it may denote the value of $y_j(*)$ for a particular argument or arguments (plugged in). Y is the set of output signal values.



(b)

CORRESPONDING QUANTITIES

EQUIPMENT (E)	MODEL (M)
TERMINALS $\tilde{u}$ $\tilde{y}$	TERMINALS u y
FAULT f_i	PARAMETER SETTINGS $\hat{P}^k$

LEGEND

Arrow ⟶ means a dependent or uncontrollable entity

Arrow ⟵ means an independent or controllable entity

"~" measured
"^" specified
" " computed

Comparison of Factors in A Posteriori and A Priori Test Specification
Figure 1.19

The interpretation that we give here to parameters is that they are variables or "quasi functions" whose value for any time t or time interval $[t - t_o]$ can be selected. In the real world, faults will occur in the equipment for reasons mentioned in Section 0.4 of Chapter 0. Knowledge of how the faults may occur physically guides the development of the diagnostic model by supplying information for formulating the fault policy. The fault simulation policy dictates the values to which the parameters must be set to simulate the equipment fault condition, f_i. In a perfect diagnostic model, the model output for a set of parameters, P^k will be the same as the equipment output with fault f_i. In this case $P^k \Rightarrow f_i$. We will sometimes denote the parameter set $\{p\}$ rather than P and the parameter value domain is denoted by Π. A particular parameter is denoted by p_i and its value by $p_i^k \in \Pi$ where again Π is the range or domain of parameter values. A particular set of parameter values will also be written $\{p_1^k, p_2^k, \ldots, p_j^k, \ldots, p_n^k\} = P^k$. *

To differentiate between ideal, assigned, measured, computed and estimated variable values or functions in Figure 1.19(b) and elsewhere, we use the following over-symbols:

$\cap$ is the ideal assigned or designated quantity

$\hat{}$ is the estimate of a quantity

$\sim$ is the measured value or values

no symbol denotes a computed value or set of values.

---

*The reader is cautioned that we will deviate from this notation in Chapter 2.

Now, returning to Figure 1.19(b), we can write

$$\underline{y}_M = \underline{y}_M(\underline{u}, P^k, t) \qquad 1.1$$

as the Model output for a given set of input terminals, input signals $\underline{u}$ applied to the terminals, output terminals equal to the dimensions of $\underline{y}$, and parameter settings P^k. Similarly

$$\tilde{\underline{y}}_E = \underline{y}_E(\tilde{u}, t) \qquad 1.2$$

is the corresponding a posteriori measured Equipment output. Equations 1.1 and 1.2 provide the data on which a test decision can be made.

1.7.1.0 A Priori Test Type Specification

The a priori Diagnostic Development Chart in Figure 1.20 shows the overall diagnostic information development process. The Testing Procedure Development (TPD), (See Figure 1.0) outlined in dashed border in Figure 1.20 is a part of the OTP area. The bold outline block within the dotted lines labelled Define Test Type represents the capability to a priori select the test type. Once the test type has been defined, the model is computed for various faults using a computer program simulation. The decision block Specify Faults Detected and/or Isolated evaluates the particular test type. Exhaustive or algorithmic methods can be used to develop a testing procedure which achieves the desired objective.

The general a priori test type will be written as the set

$$\Theta = \{IT, OT, I, O, P, \Delta\} \qquad 1.3$$

A Priori Diagnostic Development Chart

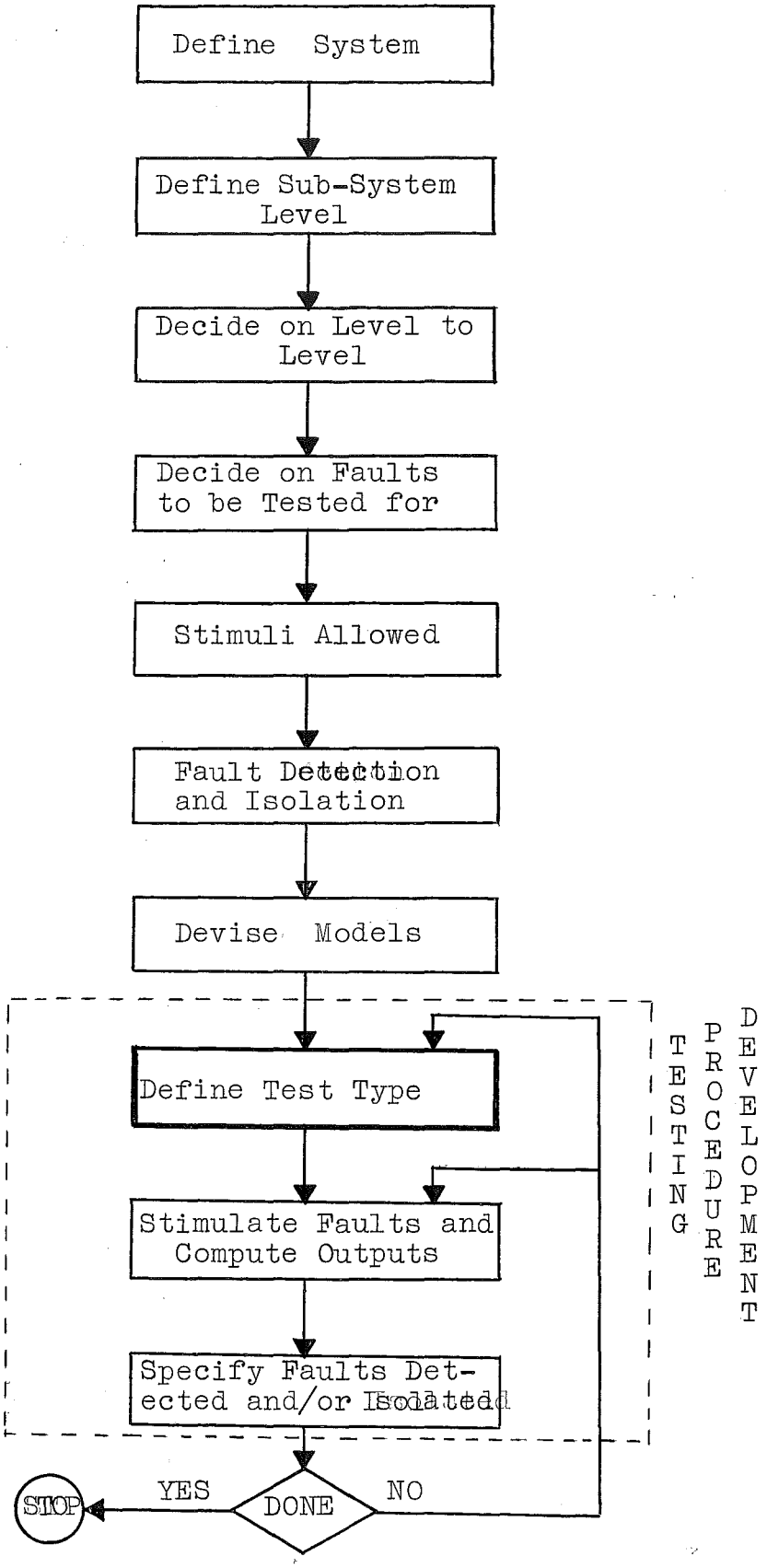


Figure 2.20

where IT = input terminals
OT = output terminals
I = input signal value (function, sequence, or symbol)
O = output signal value (function, sequence or symbol)
P = parameter value set
Δ = test decision set.

In the table shown in Figure 1.21 the test type can be

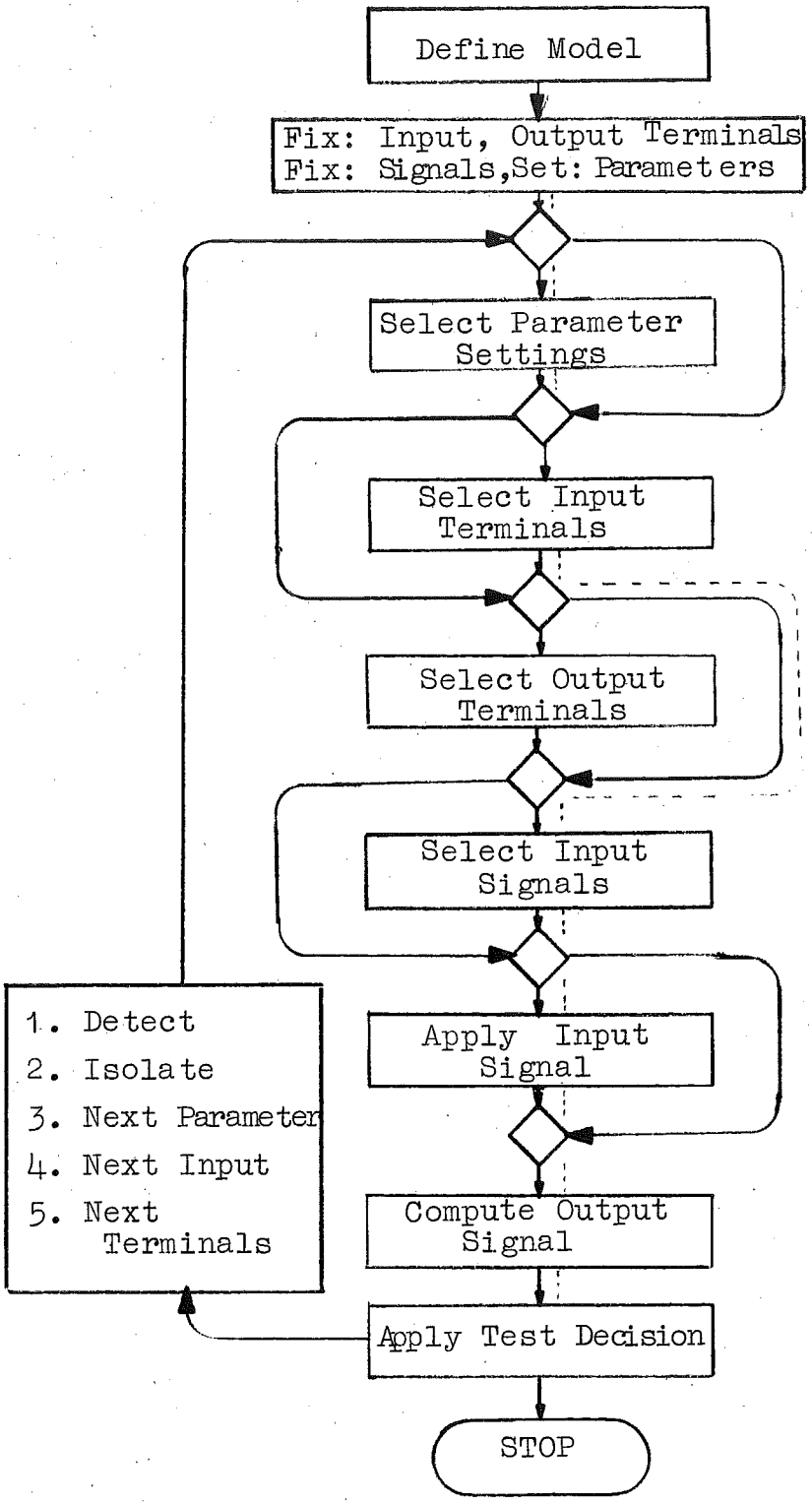
	F IT	V IT	F OT	V OT	F I	V T	F O	V O	F P	V P
TEST	a	e	b	f	c	g	-	-	d	h
Θ_1	a		b		c				d	
Θ_2	a		b			g	-	-	d	
Θ_3	a		b		c		-	-		h
Θ_4	a		b			g	-	-		h
Θ_5		e	b			g	-	-		h
Θ_6		e		f		g	-	-		h
Θ_7		e	b		c		-	-		h

LEGEND
F = Fixed
V = Variable

Test Type Table
Figure 1.21

selected by scanning along a row. Whether the quantity in the column is Fixed or Variable determines the test type. For example Θ_1 = abcd, is a fixed input terminal, fixed output terminal, fixed input, fixed parameter test type. The dash in the O columns indicates that the output is neither fixed nor variable. It is computed for a particular test. The decision set is not shown in the table.

Figure 1.22 accentuates the richness of a priori test selection possibilities. To illustrate the meaning attached to the diagram, the route traced out by test Θ_4 in Figure 1.21 has been dotted. This path corresponds to a test procedure given by the steps a,b,g,h. All a priori tests eventuate in a computation of the output



A Priori Test Type Synthesis Design
Figure 1.22

signal, once the conditions specified by Θ_i have been fixed. The diagram in Figure 1.22 is somewhat unclear in the sense that the details of how the terminals are selected and when they are selected is not specified. Nevertheless, the flexibility in selecting test types a priori is apparent. We should also mention that the Test Types shown in Figure 1.21 are only several of the possible types. However, some combinations are not as useful as others. The a priori specified test can be regarded as the procedure specification and data development stage for a posteriori testing.

1.7.1.1 A Posteriori Test Type Specification

An a posteriori test is one of the three types shown in Figure 1.18. Moore has defined the Fixed and Serial test types[107] for sequential machines. If an adaptive algorithm is used in the testing, the test will be called adaptive. This type of test has been discussed by Levadi[81].

Definition: A combinational test on a network is the quintuple

$$\{I, O, \Delta, \theta, T\}$$

where $$\theta : I \times O \rightarrow \Delta$$

and $$T : I \rightarrow I'$$

where I is the discrete or continuous input signal, O is the output signal and θ is a decision mapping of all ordered input-output pairs into the decision space, Δ. θ can also be thought of as a test result which states whether the input-output pair is indicative of a bad network or a good network. T is the rule for selecting the next input I' to be applied.

For a particular $\{i_j\} \varepsilon I$, $\{o_k\} \varepsilon O$ and $\delta_{jk} \varepsilon \Delta$, a test can be written as $\theta_{jk} : (i_j, o_k) \rightarrow \delta_{jk}$ where δ is a member of the set (good, bad, inconclusive).

In most cases, a single combinational test fails to extract sufficient information about the states of an equipment to determine whether it is good or bad. In this case, it is necessary to apply an ensemble of inputs to the input terminals and observe the resulting output. The sequence of inputs can be selected by a process which chooses the next input to apply on the basis of the present test decision. This procedure is clarified as follows: A sequential test, as its name implies, consists of a sequence of input-output pairs and a set of corresponding decisions. The decision sequence is naturally more elaborate than for the combinational test and may be used to generate next-test inputs. Following the first definition, we can add an additional rule to the test definition and obtain the following:

Definition: A <u>sequential test</u> is defined as the quintuple,

$$\{I, O, \Delta, \theta, T\}$$

where

$$\theta : I \times O \rightarrow \Delta$$
$$T : I \times O \rightarrow I'$$

where I, O, Δ and θ have their previous interpretation and now T is a mapping of the present input-output to the next input, I'.

In certain cases, it is necessary and desirable to apply an algorithm to determine what the next input signal should be. The decision may depend on the present input-output pair and on past events. The test required for this situation is termed an adaptive test. This gives

the following definition.

Definition: An adaptive test is defined by the quintuple,

$$\{I, O, \Delta, \theta, T\}$$

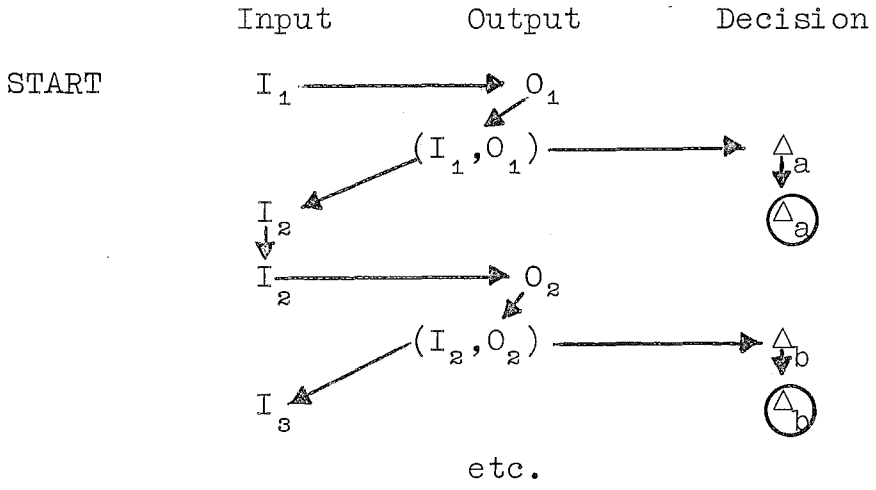
where

$$\theta : I \times O \rightarrow \Delta$$
$$T : I \times O \times \Delta \rightarrow I'$$

and I, O, Δ, θ and T have their previous definitions.

Note that the adaptive test uses a more elaborate decision mapping T for selecting the next input to apply than does the sequential test.

A compact method for illustrating the sequential test procedure is through an IOD diagram as shown in Figure 1.23 (IOD-Input-Output-Decision)



Generally,

$$(I_1, O_1) \rightarrow \Delta_1 \rightarrow (I_2, O_2) \rightarrow \Delta_2 \rightarrow I_3 \quad \text{etc.}$$

IOD Diagram

Figure 1.23

The sequential nature of the testing is clearly shown in this diagram. This pictorial method is representative of testing diagrams, test-trees and decision trees.[16,69] A diagram of the type shown in Figure 1.23 applied to fault isolation terminates when the decision is made which states which, if any, functional element is in error.

1.7.1.2 Remarks on Mappings

The definitions for test types a posteriori are given in the set theoretic-mapping language. It has been found to be a particularly useful language for expressing various ideas and relationships. We gave (in Figure 1.4) the pictorial representation of a mapping which described the operation of a type of element that accepted discrete input values and gave discrete output values. PENE and LENEF (FE) were described which perform various functions. Mappings are differentiated from functions by virtue of the property that in a mapping, the domain and codomain need not necessarily be numbers.

The discussion on diagnostic models did not dwell on the difference between memoryless and memory elements. If all elements were linear and memoryless, diagnosis could be treated solely with real number algebraic methods. (This will be discussed in more detail in Chapter 2.) Because most equipment will contain both linear and non-linear elements with and without memory, diagnostic methods must accomodate these more complex elements. The mapping is useful for expressing relationships in some nonlinear and most linear memory network models.

In a memoryless system, all of the input signals are controllable in the sense that they can be selected independently. In a system with memory, some of the

inputs, the states, are only indirectly controllable. The general mapping model which shall be used and to which we alluded in the prologue can be written

$$\Omega : U \rightarrow Y \qquad 1.4$$

where Ω is the mapping from the inputs into the outputs. U includes both independent and dependent inputs. A more usual expression for the system is

$$\begin{aligned} \sigma &: U_S \times U_I \rightarrow U_S' \\ \omega &: U_S \times U_I \rightarrow Y \end{aligned} \qquad 1.5$$

where σ is called the next state mapping, U_S is the present state values, U_S' is the set of next state values, U_I is the independent set of input values, ω is the output mapping and Y is the set of output values. The mapping representation given by Equation 1.5 can be used to model time dependent sequential discrete networks and linear differential networks. The relationship between the two has been recently discussed by Arbib[3]. Note that if U_S is empty, equation 1.4 with $U = U_I$ applies and the output is time independent, i.e. the system is memoryless.

These remarks have been inserted here for several reasons. Not the least important of these is the fact that we shall shortly use a mathematical relation in defining a test decision. Because mappings and relations resemble one another, this appeared to be a reasonable place to include this discussion.

1.7.1.3 Test-Type Comparison

The differences between model (a priori) test type specification and equipment (a posteriori) test type specification can be enumerated as follows:

1. Parameters can be controlled in a model; faults in equipment are random.
2. Model outputs are computed; equipment outputs are measured.
3. The procedure for equipment test sequencing is specified a priori. The procedure is executed a posteriori.
4. Models are used to develop decision criteria; equipment testing applies decision criteria.

These fundamental differences are reflected in test type specification and in the decision rendered by a test.

1.7.2 Test Decisions

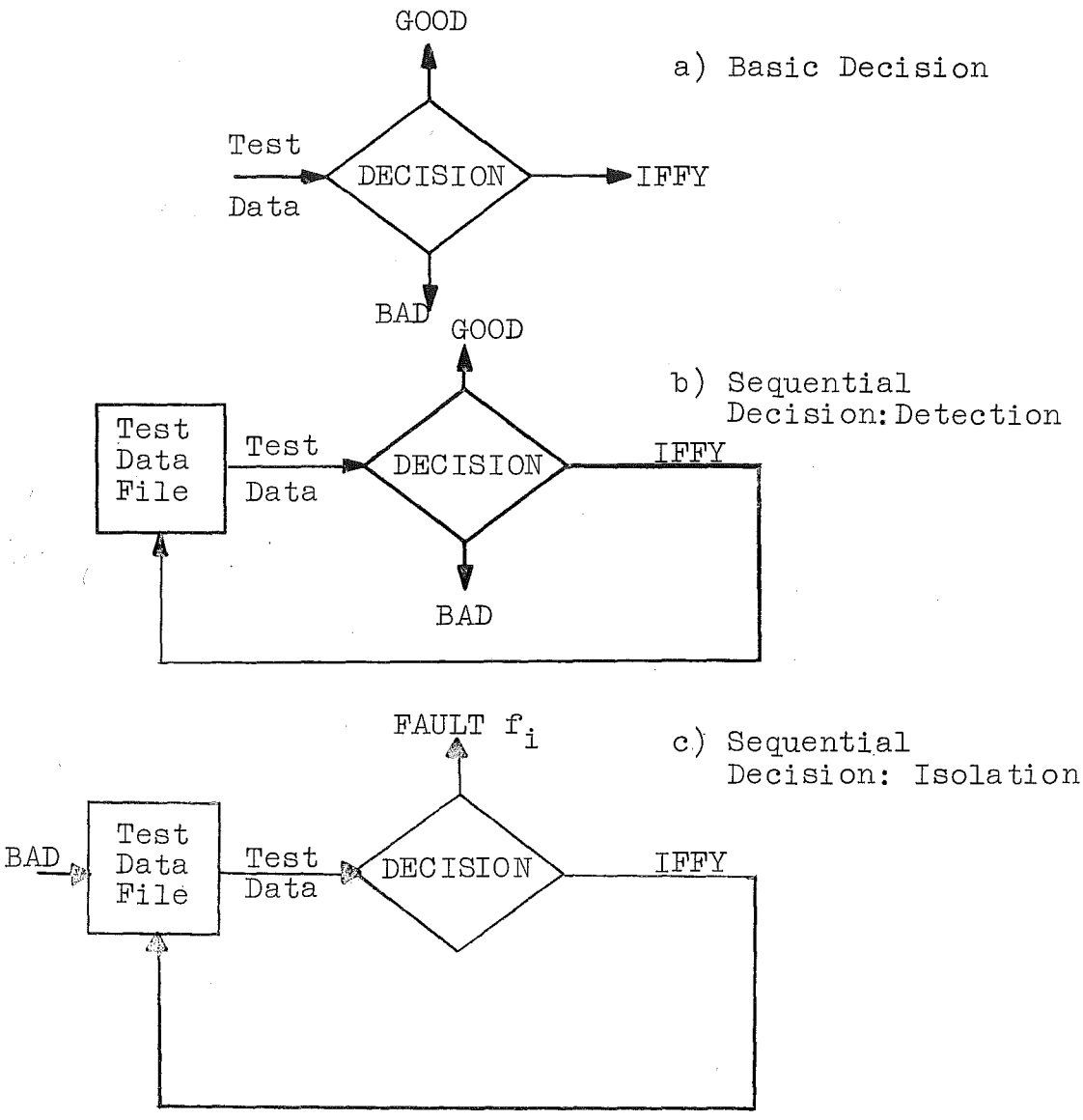
A test decision or test decision process can be used on computed data or a combination of measured and computed data to classify a model or equipment into a test decision category. If a fault policy has been stated and the total number of faults specified, then a test decision will indicate a priori whether a particular fault will be detected or isolated by a given test type. As a result of test decisions, faults which are detected (isolated) by a given test can be enumerated.

Using the results, the testability measure for a certain test type can be computed. An a posteriori test decision or test decision process uses the measured equipment outputs in conjunction with computed model output data to classify the equipment. The possible classes which have been delineated are: good, iffy, bad, fault f_i. A single test decision is usually insufficient to classify an equipment as good or fault f_i. A test decision process which uses a sequence of tests to give

a definite conclusion is required. The basic test decision problems are illustrated in Figure 1.24(a). The possible requirement for a sequence of test and decisions is indicated for detection in Figure 1.24(b) and for isolation in Figure 1.24(c). The decision category will be one of those mentioned previously. Figures 1.24 (b) and (c) indicate that IFFY category is not a "stable" conclusion. The classification category is termed a test decision conclusion.

A second and different test decision will be termed a test type decision. The test type decision blocks were illustrated in Figure 1.22. They represent decisions used in the test type selection process. The selection process may be heuristic; options are limited by constraints dictated by the physical equipment. Alternatively, algorithms executed on the digital computer can be used for the selection process. We describe several test type decision algorithms in subsequent chapters. For instance, in Chapter 2, a diagnostic method using sampled-quantized output signal data is described. The test type selection process is aided by several theorems which stipulate optimum output signal sample lengths and quantization levels. In Chapter 3, a combinational network simulation program is used to select the input pattern-output terminal combination which is optimal according to efficiency criteria discussed earlier in that Chapter. In Chapter 4, an input selection procedure, which has the objective of minimizing the number of input patterns required for fault detection in sequential networks is described.

In the remainder of this section, the test decision or test decision process will be written TD and the notation TD = {...} will denote the test decision



Single and Multistage Test Decisions
Figure 1.24

conclusion set. A test type decision is written, TTD.

1.7.2.0 Test Decisions in Fault Detection and Isolation

The following questions state in brief form, the test data conditions which are important for making a posteriori fault detection and fault isolation test decisions. As before, subscript E refers to equipment and M to model.

1. Fault Detection: Given $\underset{\sim}{\tilde{u}}_E$ and $\underset{\sim}{\tilde{y}}_E$, is the equipment fault free?

2. Fault Isolation: Given $\underset{\sim}{\tilde{u}}_E$ and $\underset{\sim}{\tilde{y}}_E \neq \underset{\sim}{\tilde{y}}^o_E$, where it is known that $\underset{\sim}{y}^o_M = \underset{\sim}{\tilde{y}}^o_E$, and $\underset{\sim}{y}^o_M$ is the good model output and $\underset{\sim}{\tilde{y}}^o_E$ is the good equipment output. Is there a $\underset{\sim}{u}_M = \underset{\sim}{u}_E$ and a $\overset{\frown}{P}^k$, k = 1,2,...,m which gives a $\underset{\sim}{y}^k_M = \underset{\sim}{\tilde{y}}_E$ for sufficient t?*

Question 1 can be answered conditionally. If $\underset{\sim}{\tilde{u}}_E$ is representative (or actually equals) all possible input signals and $\underset{\sim}{\tilde{y}}_E$ is observed to be its expected function, then the equipment can be classed as good. Otherwise, it is iffy.

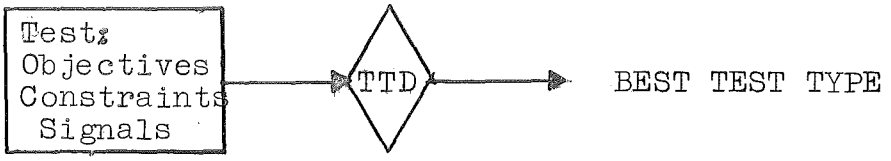
Question 2 can be restated: If the equipment is known to be bad, does one of the solutions to the model with $\underset{\sim}{\tilde{u}}_E = \underset{\sim}{u}_M$ and $P = P^k$ give the same set of output signal values as the measured set? Here successful isolation is contingent upon the goodness of the model, which of course depends in turn on the accuracy of the fault simulation policy. Overall success is dependent on such factors as noise and the correspondence between applied and computed input signals.

---

* Oversymbol characters were defined on page 1.53

1.7.2.0.0 TTD - A Priori Test Type Specification

The test type decision or test type decision process which is used in TTS and OTM studies can be illustrated as follows

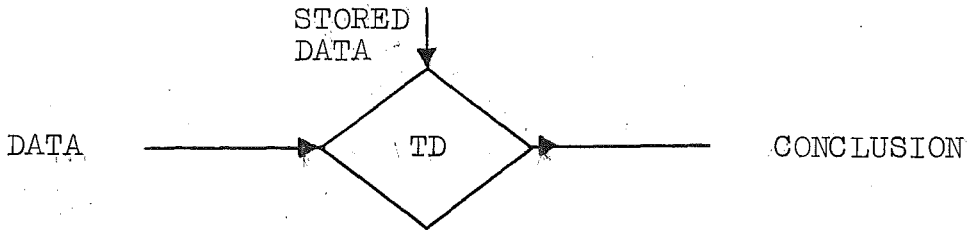


The test objectives, constraints, and signals are fed into a TTD block and the BEST TEST TYPE emerges from the opposite side. The TTD block disguises many intermediate decisions of the type illustrated in Figure 1.22. Not only must the decision blocks determine the path of a test in Figure 1.22 but each test must be followed by a computation of its effectiveness. Only when all allowable variations have been tried can the TTD block issue the BEST TEST TYPE decision. Of course, this exhaustive approach is neither desirable nor is it always feasible. Instead, alogrithms which specify locally optimal tests which are constrained either in terminal or signal selection may be developed. It is difficult to formulate these general algorithms. Consequently, many of the test procedures used in present checkout equipment are heuristic.

Several algorithms and analyst-computer program interactive methods for TTS are described in the following Chapters. They may be considered as the "guts" of the TTD decision block. Because each network class requires a particular treatment which depends on assumptions of model input signals, terminal availability, and fault type, we will not generalise on TTD at this point.

1.7.2.0.1 Combined A priori - A Posteriori Dependence

The test decision can apply either to the a priori decision about the relevance of data to a fault detection or isolation diagnostic model study or to the a posteriori problem of checkout. The basic decision activity is schematically illustrated below. In the a priori case,



data obtained from the present computation is compared with some or perhaps all of the previously computed data and the data is judged identical or different from the previous data. In the a posteriori case, the measured data is compared with good and/or faulty model-simulation computed data. A comparison of the two is used to decide whether or not the equipment and model data are equivalent. The comparison provides the mechanism for making the decision (conclusion).

In a sophisticated checkout scheme, sequential or adaptive tests illustrated in Figure 1.18 (see also Figure 1.24) are used. The sequencing of input signals will be specified by an a priori TTD process. Any allowable flexibility in selecting input and output terminals can be accommodated in the TTS.

In most previous investigations of diagnostic test development, both for analogue networks and for digital networks, a fixed input and output terminal set has been assumed[1,9]. Consequently, the TTD was limited simply to the selection of input values. (Of course the inputs will be records or sequences for analogue and sequential

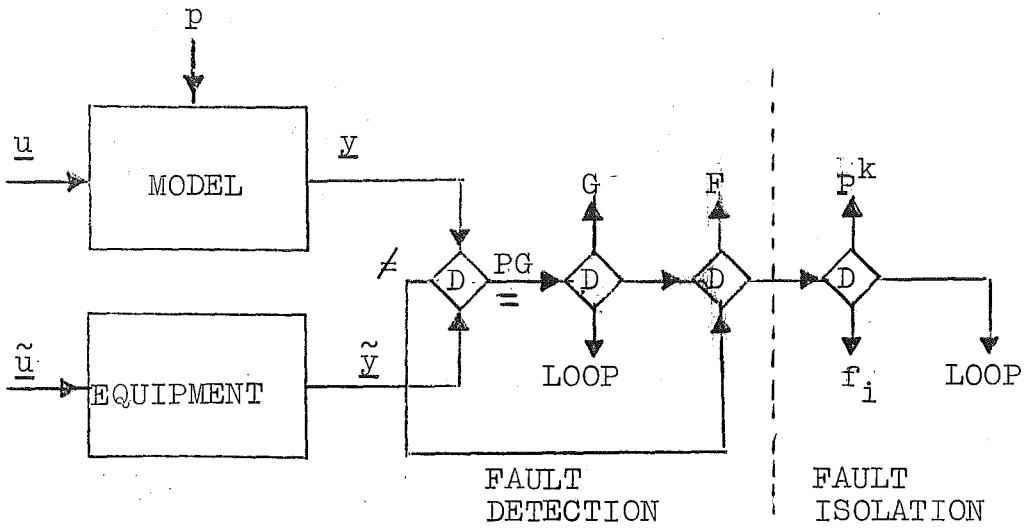
networks and patterns of bits for combinational networks.) The provision for being able to select terminals, which was introduced in TTS, can be regarded as a main difference between this approach and previously reported network diagnostic methods. What we have done is to introduce design information, in terms of a specification of the test point terminal placement, into the diagnostic development area. Rather than accepting an heuristic test point placement, the a priori TTS may actually prescribe input and output terminals and signals which will result in optimal testing.

Some of the previous discussion on test decision conclusions and the relationship between a priori and a posteriori tests will now be firmed up.

Figure 1.25 presents a diagram and table showing the relationship between the fault detection and fault isolation decisions and between the role of the model and equipment in providing information for these decision(s). In Figure 1.25 (a), the test decision diagram shows how model and equipment output information feed into a series of decision blocks labelled D_1, D_2, D_3 and D_4. A dotted vertical line is drawn to separate the fault detection and isolation decisions. In the test decision diagram, we imagine the a priori information to be developed in the upper half of the picture by the model and the a posteriori in the bottom half. Melding of the information occurs in the decision blocks which render the decisions conclusions: probably good*, good, fault, fault f_i, parameter value set P^k. The branching nature

---

*The terms "probably good", "possible good", "iffy" and "questionable" are all used to refer to a conclusion that no definite decision can be made on the evidence available.



LEGEND:
1. PG possibly good
2. G good
3. F faulty
4. f_i fault f_i
5. LOOP means "more information required for decision".
6. P^k parameter values P^k, where k=0 means good and k≠0 means fault.

Test-Decision Diagram
(a)

$\underline{u}$	P	$\underline{y}_E$	$\underline{y}_M$	Condition	Decision
$\hat{\underline{u}} = \tilde{\underline{u}}$	$\hat{P}^o$	Computed	Measured	$\underline{y}_M = \tilde{\underline{y}}_E$	D_1 : PG
$\hat{\underline{u}} = \tilde{\underline{y}}$	$\hat{P}^o$	Computed	Measured	$\underline{y}_M \neq \tilde{\underline{y}}_E$	D_2 : G D_3 ; F
$\hat{u} = \tilde{y}$	$\hat{P}^k$	Computed	Measured	$y_M = \tilde{\tilde{y}}_E$	D_3 : F D_4 : $\hat{\hat{f}}_i = P^k$

Test Decision Table
For Fixed Input Tests
(b)

Figure 1.25

of the decision block D_1 can be observed and depends on whether the two outputs are = or $\neq$. Figure 1.25 (b) is an interpretation of the diagram in terms of output signals and parameter set P given a fixed input function. The test decision table has been divided into two halves corresponding to the fault detection and fault isolation areas. One key feature of the problem of the isolation decision can be observed in the Decision column in the table. It shows that at least two decisions D_1 and D_2 must be made before the equipment can be verified as being good (G) following the observation that model and equipment outputs are equivalent. Here, the model parameter condition $P = \overset{\Omega}{P}^{o}$ and perfect accuracy are assumed. In this case, the decision may be multistage. In other words, there is a looping process required to give a G decision. Similar comments apply to the fault isolation decisions, D_3 and D_4 in the bottom half of the table in Figure 1.25(b). We will later introduce the method for executing the required decision processes by using testing diagrams[19].

It is useful to develop a test decision step or process that is independent of the type of equipment. The basic test decision step can be regarded as a binary relation[107] which classifies the test result into one of two categories. Our everyday intuitive notions of a decision provide the yes-no, will-will not, good-bad experience of decision making. In diagnosis, if ternary decisions such as yes-possibly-no, or can-might-cannot are required, the intermediate state is not considered "stable". In testing, we would like the ultimate decision to be emphatic.

1.7.2.1 Tests as Binary Relations

The mapping notation is useful for classifying diagnostic data. A mapping rule can be developed a priori using computed data. A mapping can also express the functional performance of a network. It was shown in Equation 1.4 that a memory network may require two mappings to represent it.

In both memory and memoryless networks a fault may cause the input-output behaviour to deviate. Either the input element maps to an incorrect element within the output signal range or it maps to an element outside the specified range. These two possibilities will be termed mapping errors or mismappings. An example of the first type was given in Figure 1.4. If a network mapping is time dependent, the input-output elements are a sequence of temporarily co-ordinated elements. Time independent networks form a degenerate class of the more general memory time dependent networks.

If sequences of input-output signals are obtained for good and faulty networks, these sequences can be used to formulate a test decision. Suppose that the model input-output sequences are represented by Z_M and the equipment input-output sequences are denoted by Z_E. We will suppose that a test decision can be made for each pair of signals in $Z_M \times Z_E$. If TD = {good, bad, iffy, fault f_i}, the mapping

$$\mathcal{M} : Z_M \times Z_E \rightarrow TD \qquad 1.6$$

represents the tests and subsequent decisions on the equipment E. For fault detection, comparing $\underline{y}_E$ with $\underline{y}^o_M$, given $\underline{u}$, will indicate whether the element from TD should be good, bad or iffy. For fault isolation, if for $z_M \in Z_M$

and $z_E \in Z_E$, we observe that $z_M = z_E$, then using stored data on parameter values we can determined which fault f_i exists in the equipment.

Now, Let P^k be the set of parameter values assigned to the model and $\{f_k\}$ be the corresponding equipment faults. $P^o \Rightarrow f_o$ is the good condition. We can write $f_o^c = g$ for this case. Using a model which is the estimator of the equipment, we can rewrite Equations 1.1 and 1.2 to conform to this notation as

$$\underline{y}_{\hat{M}} = \hat{M}(\underline{u}, p^k, t^*) \qquad 1.7$$

and

$$\underline{\tilde{y}}_E = E(\underline{\tilde{u}}, f_k, t^*) \qquad 1.8$$

where Equation 1.7 is the computed output and Equation 1.8 is the measured output from the equipment. t^* is discrete time. We use discrete time because in subsequent chapters, all diagnostic information on the output signals is computed or measured at discrete time intervals. Using the subscript g to denote good, we can write

$$\underline{y}_{\hat{M}_g} \triangleq \underline{y}_{E_g} \qquad 1.9$$

where we have shown a "weak equality" between the model and equipment output signals. This weak equality indicates the inevitable discrepancy between the computed and measured output signals. The inaccuracy can arise from system noise, inherent measurement errors, inexact model parameter values and computational errors in the digital computer solutions.

Equation 1.9 can be made into a stronger equality by accurate modelling. Equation 1.9 should hold not only

for the good conditions but also for fault conditions. Hence,

$$\underline{y}_{\hat{M}_{P^k}} \doteq \underline{y}_{E_{f_k}} \qquad 1.10$$

for all k, k = 0,1,2,3,...,m. To be more definite, we can write

$$\underline{y}_{M_{P^k}} = \underline{y}_{E_{f_k}} \pm \underline{\xi} \qquad 1.11$$

where the equality is strong and $\underline{\xi}$ represents the error vector. Each component of ξ is associated with a particular output terminal signal. The j^{th} terminal signal error at each sampling interval can be written as $\xi_j(t^*)$. This error may be associated with the model, the equipment or with both. One of the basic assumptions made initially was that the model behaviour should mirror the equipment behaviour. For this reason, the subject of measurement, estimation and computing errors will be discussed only briefly here. In Section 1.7.2.3, the basic questions relating the overall accuracy to test decisions are posed and discussed.

An interesting discussion on the use of an algorithm for developing a test decision from measured information is discussed by Levadi [81]. He uses component parameter drift distributions and output signal probability distribution functions to obtain information on which a test decision is made. The answers he obtains are in terms of probabilities of certain components being responsible for the observed data. Another technique reported by Chu[26] uses a somewhat similar formulation. He defines a cost function whose variables are conditional probabilities. The probabilities are for each fault type

conditioned by a certain set of observed symptoms. The observations may be noisy and the method establishes a probabilistic measure of the likely faults given the observed symptoms.

We mention these two investigations to remind the reader that the diagnostic models postulated here are ideal in the sense that the model and equipment give identical answers if both are in a noise free environment. We realize the practical limitations of such an assumption.

In applying a fault isolation technique, the chance that a P^k can be specified which will give $\tilde{y}_E = y_M$ exactly is high for digital networks. For analogue networks, the variables are stochastic. Hence, the techniques suggested by Levadi and Chu are useful if component parameter value distributions* for the equipment are available. Because the analysis can become difficult using these techniques, particularly when the network is large, methods based on discrete information have been developed in this thesis. Using discrete information, test decisions can be made without any apparent ambiguity because the decision space may itself be discrete. We say apparent, because some of the information is destroyed in the quantization process.

Although the methods of Levadi and Chu are attractive when the network is small, they can lead to intractable computations when the network gets large. In addition they depend upon a decision space whose dimensions increase with network complexity. Because the decisions

---

*These are parameter value probability distribution functions which may be discrete or continuous. Levadi assumes that the distributions are given as histogram data. Chu assumes that prior information on malfunction-symptom probabilities can be obtained.

that we will use are binary and based on discrete information, the need for parallel (analytical) computation is avoided. Using a series of simple decisions, we eliminate the requirement to compute multidimensional probability functions. We require only the simple register, word or bit comparisons for binary decisions.

A test can be defined as a binary relation. We denote the set of all measured equipment outputs by $\{\tilde{\underline{y}}\}$ and use $\{\underline{y}\}_g$ and $\{\underline{y}\}_f$ to represent all good and faulty sequences which may be measured or computed. The test decision set contains two elements. A test decision classifies the output signal information. The classification relates the information to one of the two test decision set elements. A test can be written

$$\theta = (\{\tilde{\underline{y}}\}, \{\underline{y}\}_g, \lambda) \qquad (a)$$

or 1.12

$$\theta : \{\tilde{\underline{y}}\} \times \{\underline{y}\}_g \rightarrow TD \qquad (b)$$

where θ is a relation (a) or a mapping (b) and λ is a <u>propositional function</u>[77]. The mapping θ resembles the theta discussed in a posteriori test type specification. Basically Equation 1.12 says that for any measured output sequence $\underline{y}$, if $\tilde{\underline{y}} = \tilde{\underline{y}}_g \varepsilon \{\underline{y}\}_g$, a decision conclusions will be one of the elements in TD; otherwise, it will be the second element. To make this more definite, we can let TD = {good,bad}. Then if $\tilde{\underline{y}} = \underline{y}_g$, the element from TD is "good". If $\tilde{\underline{y}} \neq \underline{y}_g$, the element from TD is "bad". Or,

$$\tilde{\underline{y}} \varepsilon \{\underline{y}\}_g \Rightarrow E \quad \text{good}$$

$$\tilde{\underline{y}} \notin \{\underline{y}\}_g \Rightarrow E \quad \text{bad}$$

where ε is the sequence or set membership symbol. $\notin$ means

"is not a member". It is instructive to write one of the basic definitions for the binary relation.

> Definition: A relation ρ is the sets $\mathcal{A}_1$ and $\mathcal{A}_2$ and the propositional function[82] $\lambda(\nu,\eta)$. $\lambda(a,b)$ is either <u>true</u> or <u>false</u> for any ordered pair (a,b) belonging to $\mathcal{A}_1 \times \mathcal{A}_2$. i.e., $a \varepsilon \mathcal{A}_1$ and $b \varepsilon \mathcal{A}_2$.

If TD in Equation 1.12 consists of the set {TRUE,FALSE} then if $\lambda(a,b)$ is true, we write:

$$a \rho b \quad \text{(a related to b)}$$
$$\text{otherwise} \quad a \not\rho b \quad \text{(a not related to b)}.$$

For fault isolation, let $\{y\}_{b_1} \cap \{y\}_{b_2} = \{y\}_f$, "$\cap$" means intersection and b_1 and b_2 denote the fault classes got by partitioning the fault class f into two sets. A fault isolation test decision can be treated as a binary decision, sequential in this case, and using TD = $\{b_1,b_2\}$ if we write

$$\begin{aligned} &\text{given} && \tilde{\underline{y}} \not\varepsilon \{\underline{y}\}_g, && \text{(a)} \\ &\text{then} && \tilde{\underline{y}} \varepsilon \{y\}_{b_1} && \text{(b)} \\ &\underline{\text{or}} && \tilde{\underline{y}} \not\varepsilon \{y\}_{b_1} \Rightarrow \tilde{\underline{y}} \varepsilon \{y\}_{b_2} && \text{(c)} \end{aligned} \qquad 1.13$$

By successive partitioning, we obtain

$$\tilde{\underline{y}} \varepsilon \{\underline{y}\}_f, \tilde{\underline{y}} \varepsilon \{\underline{y}\}_{b_i}, \tilde{\underline{y}} \varepsilon \{y\}_{b_j}, \ldots, \tilde{\underline{y}} = y_{Pk} \Rightarrow f_k \qquad 1.14$$

where $f_k \varepsilon b_j \subset b_i \subset f$. $\subset$ means "contains" .

Equation 1.14 indicates that successive partitioning of the set $\{\underline{y}\}_f$ finally leads to a singleton. This gives a fault isolation decision. In the next section, we illustrate the graphical version of Equation 1.14, called a testing diagram. Sequential tests are now summarised.

For the test decision set TD = {GOOD, BAD} and for the measured and/or computed $\tilde{y}$ and y_G and y_{P^k}, tests based on fixed input signals are summarised in Figure 1.26.

Test Data - Decision Table

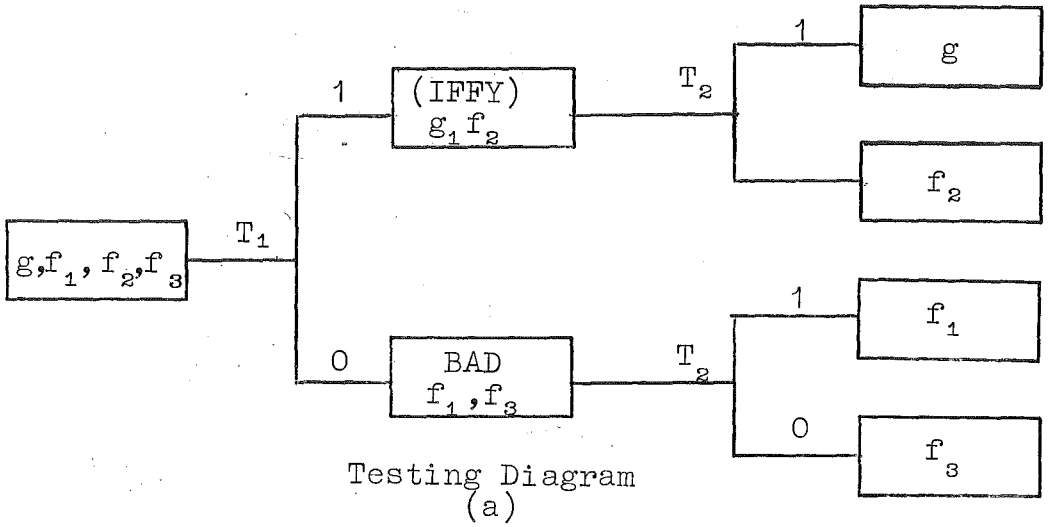
No.	Condition	Decision	Remark
1	$\tilde{\underline{y}} = \tilde{\underline{y}}_G$	Good	$\tilde{\underline{y}}_G$ obtained from good equipment
2	$\tilde{\underline{y}} \neq \tilde{\underline{y}}_G$	Bad	Equipment contains fault
3	$\tilde{\underline{y}} = \underline{y}_G$	Good	Equipment output and model output identical
4	$\tilde{\underline{y}} \neq \underline{y}_G$	Bad	Assuming network is accurate
5	$\tilde{\underline{y}} = \underline{y}_{P^k}$	Bad	Equipment fault correponds to model parameter set P^k
6	$\tilde{\underline{y}} \neq \underline{y}_{P^k}$	?	Equipment may be good or bad

Figure 1.26

1.7.2.2 Test Information Display

Fault detection and isolation can be treated simultaneously using information in the form of a testing diagram or fault table. A single TD can be regarded as reducing the uncertainty in the information about the particular P^l or about the status of the equipment.

The testing diagram shown in Figure 1.27(a) is representative. Each block lists the possible condition of the equipment after a test (or series of tests) has been applied. In the diagram, two tests, T_1 and T_2 are required to ascertain that the network is good or that one of three faults exists. A 1 is placed on a branch



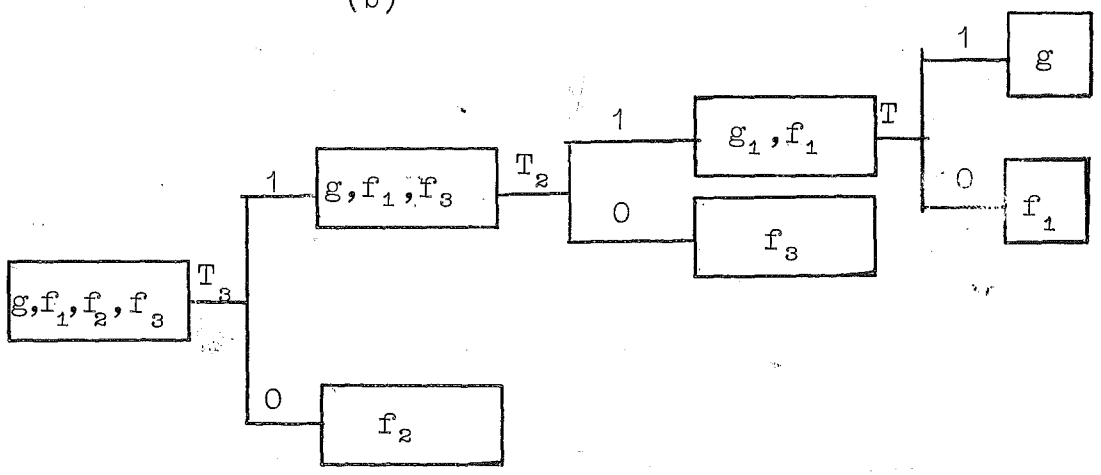
Testing Diagram
(a)

TEST	g	f_1	f_2	f_3
T_1	r_1	0	1	0
T_2	r_2	1	0	0
T_3	r_3	1	0	1

Fault Table
(b)

LEGEND:
1 = r
0 ≠ r

A 1 appears in the row-column intersection if the result r for fault f_i is equal to the good result



Testing Diagram
(c)

Figure 1.27

leaving the junction of the diagram to denote that the good and faulty networks give identical results. If the results differ, a "0" is used. This diagram is contrived. Some tests may result in empty blocks which are equivalent to inconclusive results. Another factor not evident is the non-uniqueness of the diagram, the order in which the tests are applied need not be the same as shown. This can be observed by referring to the fault table in Figure 1.27(b). If for example T_3 is applied first, the testing diagram shown in Figure 1.27(c) results. Note that it requires more steps than does the diagram in Figure 1.27(a).

We shall apply testing diagrams and fault tables to develop fault detection and isolation information in subsequent chapters.

1.7.2.3 Test Decisions and Accuracy

We now interject several points on the question of accuracy. A full discussion on diagnostic information accuracy is not proposed. Rather, we reveal some of the very basic questions which should not be overlooked when discussing the theoretical development and practical application (a priori) of the test methods.

Point 1. Digital networks are not nearly as susceptible to signal value accuracy as are analogue networks.

Point 2. Quantized analogue signal information can reduce the requirement for high computational and measurement accuracies when developing and applying diagnostic testing methods.

Point 3. There are four basic errors that are introduced at various stages of the diagnostic process. These are

a) computing noise
b) model inaccuracy
c) fault simulation policy inaccuracy
d) measurement noise.

These points must be carefully considered when developing diagnostic information. One important technique for dealing with problems of inaccuracy is to use component failure data as we have previously mentioned. But because the basic precept of the analytical approach is prediction and because diagnostic information is difficult to verify, accuracy in all aspects of diagnostic development is essential. The outcome of a test decision can only be as reliable as the information on which it is based.

In the following section an example is given which quantitatively illustrates the philosophy and basic methodology of fault detection and isolation in the system environment.

1.8 EXAMPLE

This example is presented to illustrate and amplify some of the important ideas presented so far in Chapter 0 and 1. In particular, an interface network obtained from an equipment partition will be modelled using connected FE.'s. Using a time domain method which samples and quantizes the output signal, information for fault detection and isolation is developed.

Equipment

The electrical equipment is assumed to be part of a larger analogue digital system such as a process controller. The non-electrical parts are separated from the purely electrical components by transducers. A partitioning of the equipment specifies pure analogue, combinational, and sequential networks and several interface networks. The interface network equipment diagram for the network to be analysed is shown in Figure 1.28 as EXAMPLE MEMORYLESS SYSTEM DIAGRAM. Included in the figure are its name, inputs and FE parameters, P(1) through P(11). Signals are shown as S(I) and Y(I).

FE Diagnostic Model

The diagnostic model can be developed from the equipment diagram. The FE symbols are interpreted as

+ ≡ Summer

C ≡ Comparator

X ≡ Multiplier

G ≡ Amplifier

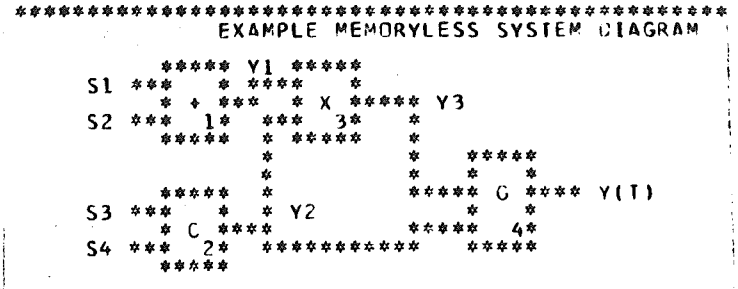
It is assumed that each of the four FE are integrated circuit units. Therefore, the PENE model is not warranted. The following fault policy is used in developing reduced parameter FE models:

1) Fault Policy

We will assume that

a) Input signals are measured perfect but can be degraded after measurement points.

b) Summers fail due to change in gain.




```
NONLINEAR MEMORYLESS NETWORK

S1,S2,S3,S4=INPUTS

P(1)=FREQUENCY(1), P(3)=FAULT S1, P(5)=FAULT S3
P(2)=FREQUENCY(2), P(4)=FAULTS2, P(6)=FAULT S4
P(7)=FAULT(+,1), P(9)=SET=1, FAULT(C,2,HIGH)
P(8)=FAULT(C,2,ZERO), P(10)=SET=1, FAULT(*,3)
P(11)=FAULT(G,4),SET=1
```

NETWORK

PNOM(1) =	20000.0000	PMIN(1) =	19990.0000	PMAX(1) =	20010.0000
PNOM(2) =	3000.0000	PMIN(2) =	2000.0000	PMAX(2) =	4000.0000
PNOM(3) =	1.0000	PMIN(3) =	0.0	PMAX(3) =	1.5000
PNOM(4) =	1.0000	PMIN(4) =	0.0	PMAX(4) =	1.5000
PNOM(5) =	1.0000	PMIN(5) =	0.0	PMAX(5) =	1.5000
PNOM(6) =	1.0000	PMIN(6) =	0.0	PMAX(6) =	1.5000
PNOM(7) =	1.0000	PMIN(7) =	0.1000	PMAX(7) =	2.0000
PNOM(8) =	1.0000	PMIN(8) =	0.0100	PMAX(8) =	2.0000
PNOM(9) =	1.0000	PMIN(9) =	0.1000	PMAX(9) =	2.0000
PNOM(10) =	1.0000	PMIN(10) =	0.0010	PMAX(10) =	2.0000
PNOM(11) =	1.0000	PMIN(11) =	0.9000	PMAX(11) =	1.1000

```
NUMBER OF PARAMETERS =11
NUMBER OF OUTPUT SAMPLES =50
SAMPLING INTERVAL =      0.0010SECONDS
```

PARAMETER VALUES

```
7 A=P(1)*T
  S1=.5*COS(A)*P(3)
  C=2*A
  S2= 1.*(COS(C))*P(4)
  S3=4.*P(5)
  B=P(2)*T
  S4=12.*COS(B)*P(6)
  Y1=(S1+S2)*P(7)
  Y2=0.+(P(8))
  IF(S3.GE.S4) Y2=10.*P(9)
  Y3=Y1*Y2*P(10)
  Y(ITER)=200.*(Y3+Y2 )*P(11)
  ********************************
```

EQUATIONS
(FORTRAN)

Figure 1.28

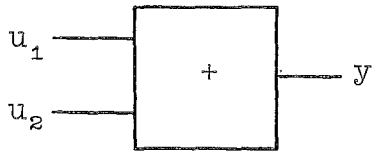
c) Comparator faults are due to changes in firing level or output level drift.

d) Multiplier faults result in inaccuracies at the output or variations in signal level.

e) Amplifier faults manifest themselves as changes in gain or apparent input signal values.

2) Model Expressions

The integrated circuit functional elements in reduced parameter form are shown below with their appropriate expressions and parameter value ranges.

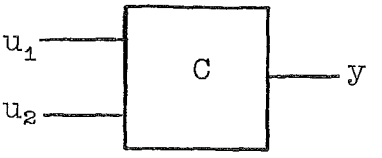


$$y = (u_1 \cdot p_1 + u_2 \cdot p_2) \cdot p_3$$

$0 \leq p_1 \leq 1.5$

$0 \leq p_2 \leq 1.5$

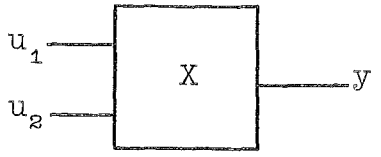
$0 \leq p_3 \leq 1.5$



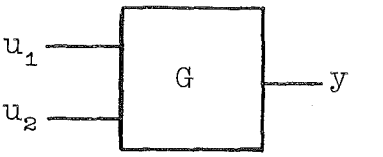
$$y = \begin{cases} 10 \cdot p_1 & u_1 \cdot p_2 \geq u_2 \cdot p_3 \\ 0 + p_4 & u_1 \cdot p_2 < u_2 \cdot p_3 \end{cases}$$

$.1 \leq p_1 \leq 2 \qquad 0 \leq p_3 \leq 1.5$

$0 \leq p_2 \leq 1.5 \qquad 0.01 \leq p_4 \leq 2$



$$y = (u_1 \cdot p_1 \cdot u_2 \cdot p_2) \cdot p_3$$



$$y = (u_1 p_1 + u_2 p_2) \cdot p_3$$

$0.9 \leq p_3 \leq 1.1$

$p = 1$

$p = 2$

3) Computer Coding

The parameter values selected are coded as PNOM(I), PMIN(I) and PMAX(I). From the network diagram shown in the top and middle sections of Figure 1.28, the FORTRAN equations can be developed. These are shown in the bottom diagram of Figure 1.28. The input signals are:

$S1 = 0.5 \cos(2\times10^4 t)$ millivolts
$S2 = \cos(4\times10^4 t)$ millivolts
$S3 = 4$ millivolts
$S4 = 12 \cos(3\times10^3 t)$ millivolts

The input signals have fault simulation parameters appended. These are shown in the FORTRAN equations in Figure 1.28. The output solution for the network output voltage, computed at time instants 0.1 millisecond apart is shown in Figure1.29. (All voltages are in millivolts). This will be referred to as the solution YGOOD(T) written in FORTRAN notation. The solution is highly non-linear and the lines drawn between the points are only given as a guide to follow the solution trajectory.

4) Parameter Variation - PARVAR (140 FORTRAN STATEMENTS)

A computer program entitled PARVAR was written to perform one-at-a-time variations in each parameter and to compute the solution for each parameter value set. Each parameter is varied incrementally from its minimum to maximum value and the output solution is computed. All parameters except the one being varied, are set to their design values designated PNOM(I) in Figure 1.28. For each solution the difference between the solution obtained and YGOOD(T) is computed and plotted. When all parameters have been varied, a summary is plotted.

Figure 1.29

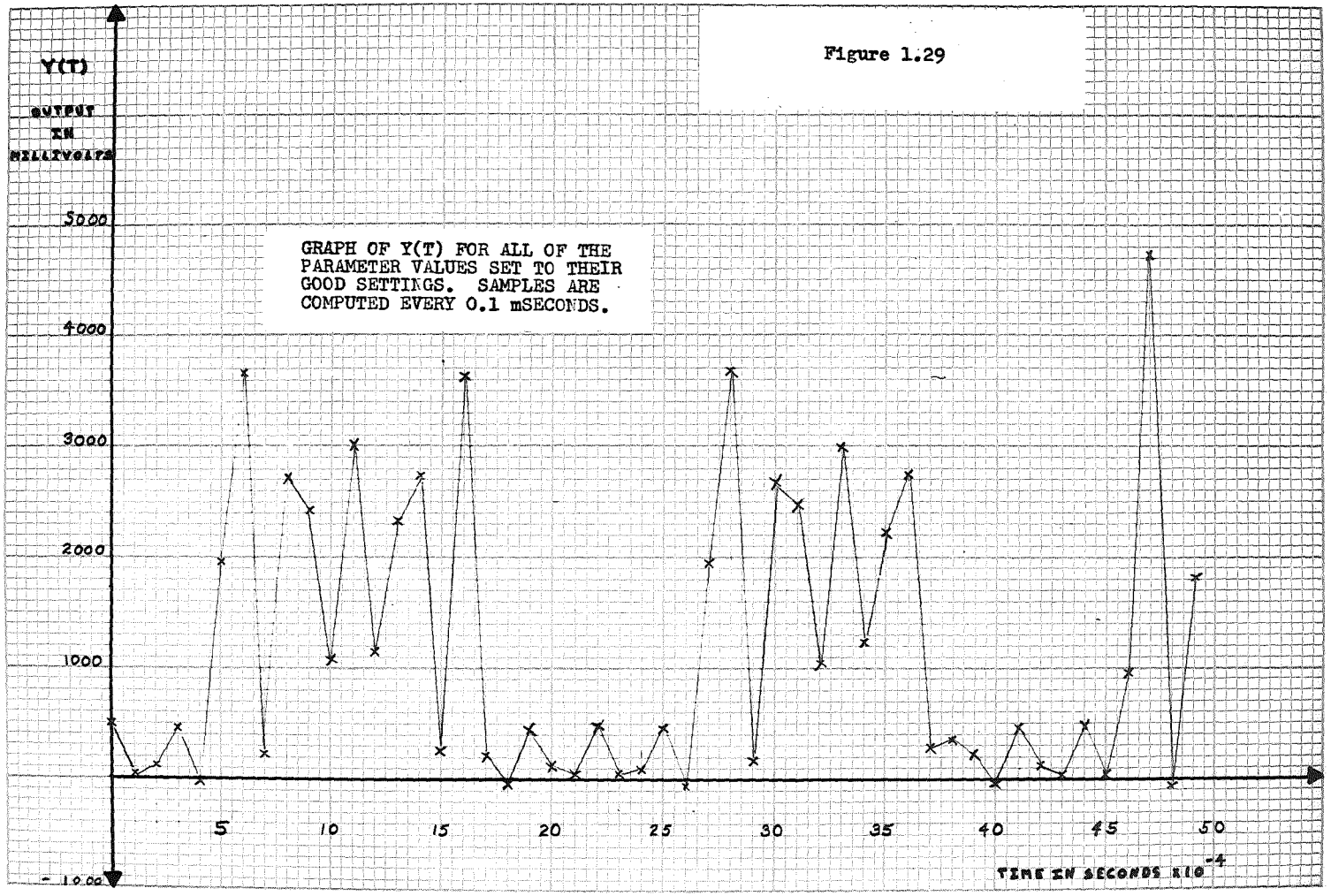

Y(T)
OUTPUT
IN
MILLIVOLTS
5000
4000
3000
2000
1000
-1000
GRAPH OF Y(T) FOR ALL OF THE PARAMETER VALUES SET TO THEIR GOOD SETTINGS. SAMPLES ARE COMPUTED EVERY 0.1 mSECONDS.
5
10
15
20
25
30
35
40
45
50
TIME IN SECONDS X10 -4

Figure 1.30 shows the output difference YDIFFERENCE during each interval of time. Intervals are the same as those in Figure 1.29, (0.1 msec.). In this plot, parameter 9 has varied from 0.1 to 2 in increments of 0.02. From the network and equations in Figure 1.28, this variation can be seen to correspond to a change in the "HIGH" comparator gain. As shown underneath the network diagram, P(9) is ordinarily set equal to 1. Setting it to values other than 1 causes Y(2) to deviate when the comparator is"ON" i.e., the output is high or nominally 10 volts. The left hand column of Figure 1.30 gives the actual difference between YGOOD(T) and Y(T) with P(9) = 0.1. The graph scale is approximately 0.085 mvolts per division. Consequently, we are assuming that the quantization levels are 0.085 millivolts apart. This is the resolution that one must be able to measure output data too. The sampling intervals are 0.1 msec apart and are exact. In Figure 1.30, an asterisk (*) is placed in the quantization level if one of the solution values YGOOD(T) - Y(T) falls within that particular level. For example, in Figure 1.30, consider the sample taken 0.7 msec after the start. Values of 0.2116 mvolts $\pm$.17 mvolts occur for values of P(9) between 0.2 and 2. Because the comparator output is "low" between time 1.7 msec and 2.5 msec after the start, P(9) has no effect on the output solution. By adding the values in Figure 1.30 for both positive and negative output differences to the plot in Figure 1.29, and envelope of the output values which will occur for various comparator gains can be obtained. For fault detection, any value of the output signal which falls outside the value YGOOD(T)$\pm$0.085 mvolts can be detected.

Fault isolation can be achieved by applying the serial TD process or possibly by observing the values of

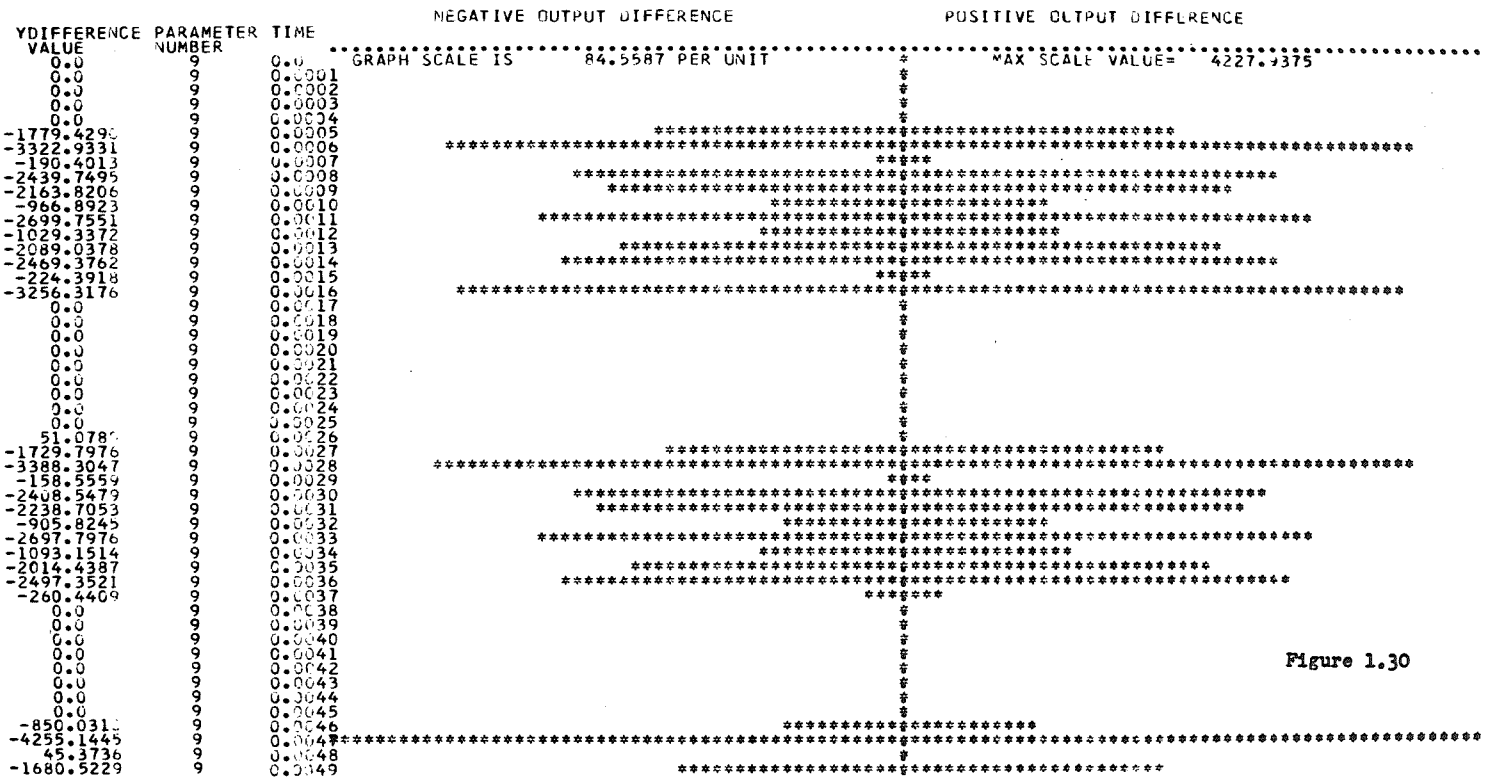

NEGATIVE OUTPUT DIFFERENCE
POSITIVE OUTPUT DIFFERENCE
YDIFFERENCE VALUE
PARAMETER NUMBER
TIME
GRAPH SCALE IS 84.5587 PER UNIT
MAX SCALE VALUE= 4227.9375
0.0 9 0.0
0.0 9 0.0001
0.0 9 0.0002
0.0 9 0.0003
0.0 9 0.0004
-1779.4296 9 0.0005
-3322.9331 9 0.0006
-190.4013 9 0.0007
-2439.7495 9 0.0008
-2163.8206 9 0.0009
-966.8923 9 0.0010
-2699.7551 9 0.0011
-1029.3372 9 0.0012
-2089.0378 9 0.0013
-2469.3762 9 0.0014
-224.3918 9 0.0015
-3256.3176 9 0.0016
0.0 9 0.0017
0.0 9 0.0018
0.0 9 0.0019
0.0 9 0.0020
0.0 9 0.0021
0.0 9 0.0022
0.0 9 0.0023
0.0 9 0.0024
0.0 9 0.0025
51.0782 9 0.0026
-1729.7976 9 0.0027
-3388.3047 9 0.0028
-158.5559 9 0.0029
-2408.5479 9 0.0030
-2238.7053 9 0.0031
-905.8245 9 0.0032
-2697.7976 9 0.0033
-1093.1514 9 0.0034
-2014.4387 9 0.0035
-2497.3521 9 0.0036
-260.4409 9 0.0037
0.0 9 0.0038
0.0 9 0.0039
0.0 9 0.0040
0.0 9 0.0041
0.0 9 0.0042
0.0 9 0.0043
0.0 9 0.0044
0.0 9 0.0045
-850.0313 9 0.0046
-4255.1445 9 0.0047
45.3736 9 0.0048
-1680.5229 9 0.0049

Figure 1.30

a single output record. Figure 1.31 shows a SUMMARY OF RESULTS. Each parameter has been coded by a symbol. These are shown in the top left of the figure. If two or more output values lie in the same difference interval, a plus (+) sign is entered. Parameters which give unique output indications are: 2,4,6,8,9,10. Parameter 9 causes the most unique output variation. This can be observed by the frequency of occurrence of the 9's.

If the output signal for any parameter setting traces out a difference path in Figure 1.31 which intersects pluses only, then the YDIFFERENCE plots for all the parameters are required. At each time interval, the parameter value settings which give the output can be ascertained. By applying the serial process decision implied by Equation 1.14, it may be possible to uniquely identify the fault. We will discuss this possibility at greater length in Chapter 2.

1.9 SUMMARY AND CONCLUSIONS

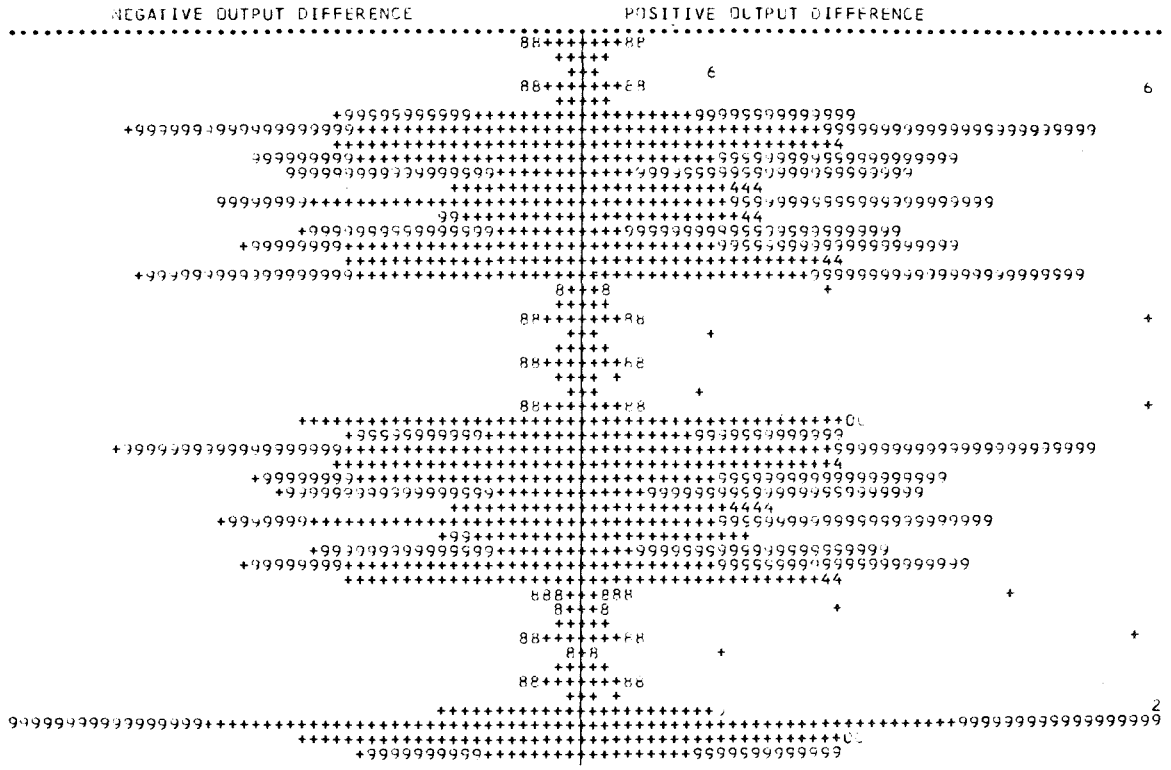
In this chapter, we have traversed the hybrid electrical system gamut. A brief historical resume was presented and important contributions in the literature which have influenced the development of diagnosis in systems were cited. Contributions from such fields as medical diagnosis, detection theory and decision theory have not been covered. Certainly however, there will be overlap between the various areas and a more comprehensive study could be undertaken to establish the relationships. This would be of mutual benefit to all disciplines concerned with the general problem of diagnosis and estimation.

SUMMARY OF RESULTS

NUMBER OF PARAMETERS=11,NUMBER OF SAMPLING INTERVALS=50,NUMBER OF PARAMETER STEPS=[illegible],TIMESTEP=[illegible]100

P(1)=1
P(2)=2
P(3)=3
P(4)=4
P(5)=5
P(6)=6
P(7)=7
P(8)=8
P(9)=9
P(10)=0
P(11)=-

Figure 1.31



The basic premise established in this chapter is that models for electrical networks can be formulated which are useful for diagnostic studies. These studies apply the models to predict the performance of a network both with and without malfunctions. Data computed from these models can be used to develop fault dictionaries or the equivalent data files for checkout equipment. Another important model application is developing efficient and effective testing methods. The development of these methods depends on availability of input-ouput signal and fault condition information on the network to be tested. Because manual fault simulation methods are tedious, expensive and possibly ineffective, the use of models is attractive. Models also have the advantage of being flexible.

If diagnosis is to be viable in a large system, it must be "a part" of the system. Consequently, the off-line testing techniques on both analogue and digital networks have been ruled out here. All techniques proposed are essentially on-line time domain diagnostic techniques.

The activity of diagnosis is directed to making decisions about the process which has generated a set of data. In this case, the process is an equipment or model and the data is input-output signals. A test decision a priori would be to estimate which of a set of parameter values should be associated with a model given a set of data. A test decision a posteriori is used to determine if the equipment data indicates that a fault exists and if so, which one it is.

In Section 1.7.1, a process was described which specifies a test type. The test type specification process is complex and warrants further investigation.

Because both the test data and the order in which they are obtained are important, an optimal test type must consider a wide range of possibilities. Many test procedures can be only sub-optimal because they inhibit the selection of terminals. Test terminals are generally specified by the network design engineer and may or may not be useful for a particular test type.

CHAPTER 2 PAGE NO.

2 ANALOGUE NETWORKS

2.0 INTRODUCTION

The primary aim of this chapter is to develop algorithms for a priori fault detection and isolation in analogue systems. In particular, network models of analogue systems will be used in conjunction with parameter variation techniques to compute tests which are efficient and thorough. Analogue networks is the generic term for the class of differential systems which admit to lumped parameter modelling. Methods which use quantized signal space techniques in the time domain for test development and evaluation will be elucidated.

Most of the analogue network diagnostic techniques reported in the literature are based on frequency domain techniques.[13, 130] Because the operating environment of many present day systems precludes the use of sinusoidal or other external stimulation, methods which use actual operating signals for test purposes are very attractive. Levadi[84] has described a method for using time domain signals to diagnose faults in a non-linear analogue network. He applies a learning algorithm in conjunction with parameter value distributions to isolate faults. There are, in addition, a number of papers on time domain techniques for system identification which have appeared in the literature[68, 105, 103] but these are not particularly relevant to the problem of diagnosis. In diagnosis, it is assumed that network parameters are known initially. Methods for determining when a parameter has changed, which parameter has changed and to what value are desired. In identification, the emphasis is more on estimating model parameters where the model is either in transfer function or sometimes in differential equation form.

There are many references on automatic checkout techniques which employ empirical methods for testing a system[147]. The spirit of this thesis is to develop methods which can be quantitatively evaluated for their testability measure. Most automatic checkout methods are not amenable to this sort of evaluation.

Perhaps the reference most relevant to the material presented in this chapter is a paper by Valstar[141]. He demonstrates that time domain techniques can be used for transfer function tracking[144]. Using filtering methods in the time domain in combination with digital computation, he shows that it is possible to solve for the network parameters using input-output information only. However, his method requires the inversion of large matrices and the number of filters required to obtain the data goes up as the square of the number of parameters.

The very thorough treatment of linear systems by Zadeh and Desoer[148] and the work of Wymore[145] and Lofgren[83,84] are also pertinent to the approach taken in this chapter.

2.0.0 Approach

The ideas propounded by Zadeh and Desoer on linear systems (see in particular Chapters 1-3), will be applied "heuristically". We regard the analogue network as an oriented abstract object[148] and subsume Zadeh's ideas of state, input and output in concept (if not in detail). A network model will not be defined as a relation however. For conceptual purposes, the mapping definitions developed in Chapter 1 are employed so as to maintain a certain consistency in the material.

In the discussion in this chapter, it is assumed that faults are due to parameter changes and that they occur one-at-a-time. To eliminate any extraneous sources of error, we also assume that the initial state (initial conditions) canalways be specified, measured, or computed and that the input to the network is obtained from some error free source and is prespecified and invariant for a class of tests. Furthermore, time will be reckoned from an arbitrary "zero" and all system signals are analogue. Basic to the methods developed in Sections 2.6-2.9 is the assumption that the output signal is measured by sampling at discrete times and that the value of the output signal is quantized. Using quantized signal information, a class of fault detection and isolation methods is developed.

The linear networks which we will be discussing can be modelled by linear differential equations. Non-linear networks should be expressed in solved form. At the inception of this study, it was hoped that simulation of faults in linear networks could be achieved by computing solutions for networks with time varying parameters using state space techniques. The idea was to be that by fixing the argument of the parameters of a time varying system, it should be possible to simulate a conventional fixed parameter network whose parameters had changed due to drift errors. This approach was not pursued because it represented no advantage over simply varying the parameters by simulation methods. In fact, it is not as direct as the variation methods described here if parameters change in some complicated fashion or if they change very slowly with respect to real time.

Moreover, the state space method comes off second best if non-linear components are predominant in the network. Nevertheless, for computer solutions, state space models can be employed to advantage in many cases.

In most of the subsequent discussion, a network model parameter change and its physical system counterpart will not be differentiated. In essence the words fault, error, degradation fault and parameter perturbation all refer to corresponding phenomena. It will usually be clear when a distinction between a fault in the actual system fault and a change in the network model parameter values should be made.

2.0.1 Outline of Objectives and Results

Our first objective will be to describe the class of analogue network models which is useful for diagnostic studies. Using the d.c. network model, we illustrate the difficulty of analytically determining the parameter values of the network using output and input signal information only. The analogue network can be regarded as a memory which stores the parameter value information. The output signal contains information about the parameters in a coded form. The difficulty of determining parameter values indirectly follows from the inability to "address" a parameter value directly, (thinking here in terms of the memory analogy).

It has been shown ([13], [14], [1]) that indirect determination of parameter values in certain network configurations is feasible. For example, Berkowitz [13] gives the conditions for being able to determine parameter values knowing the component topology. His method specifies the number and types of terminal measurements which are required to find the piece part values. We do not pursue this approach because the class of networks to which the

Berkowitz conditions apply is too limited to be of great practical significance.

The second objective is to demonstrate the use of sensitivity (See [35]) as a measure for estimating the effect of parameter changes on a given output. It follows from the discussion of sensitivity that if a parameter is "addressable" through a given output, the output is useful for detecting variations in the parameter. The inference is that a fault detection test will require a number of outputs sufficient to detect changes in all parameters and that the effect must be sufficiently large enough to measure. The reader may sense the parallel between "addressability" and observability.[35]

The third and major objective of this chapter is to develop a class of methods for the synthesis of fault detection and isolation tests. The methods depend on quantized and sampled output information obtained from network model simulations. This class of technique for analogue test development has not been reported previously. However, we assert that this new approach represents an attractive method for a priori test specification which results in a posteriori methods, viable in the integrated system environment.

A tacit assumption for the studies performed here is that diagnostic information should be generated and presented in a form which ultimately can be used to develop an on-line automatic test method or a fault dictionary[147] which may be used either off-line or on-line in conjunction with a general purpose digital computer.

To summarize, the general requirements for analogue network fault detection or isolation techniques are:

1. The a priori diagnostic information is generated off-line using output signal information from the network model. Output signals are analysed in discrete form. The network must be modelled in a form which permits one-at-a-time parameter perturbations to be simulated.

2. The actual diagnostic process should be run on-line in conjunction with the information created in step 1 and should involve the minimum interruption in the actual physical system. Ideally, the actual system signals should be used.

3. The information generated for diagnostic purposes should be condensed as much as possible and should be useful for both automatic and manual diagnostic applications.

2.1 ANALOGUE NETWORK DIAGNOSIS

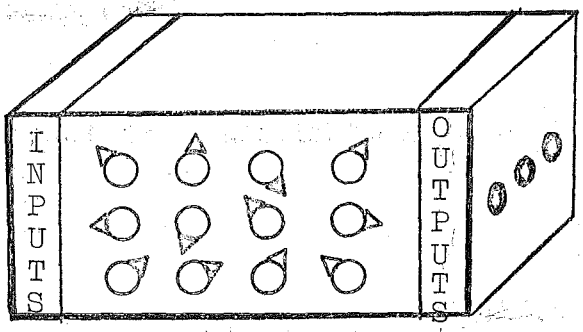
To obtain diagnostic information, a network model or physical simulation of the equipment is developed in which the model parameter values are made to correspond to the actual component functional element or piece part values. By manipulating both the model inputs and parameters and by observing the model outputs, tests can be specified for fault detection and isolation in the physical system.

It is useful to know if the form of the model will allow all faults to be detected and isolated by a particular test. In order to specify the testability measure for a model-test procedure combination, certain fundamental restrictions contingent on the network formulation will be presented. The conditions must be related to the basic fault definitions given in Chapter 1.

In essence, these conditions state that if a component parameter change causes a mis-mapping to occur, then it is possible to detect the fault by comparing the fault free system output with the output from a network model containing the particular fault. If the mis-mapping is unique for this condition, the output signal will be different from all other possible output signals. In this case, isolation of the fault is possible.

2.1.0 Inputs, Outputs, Parameter Variation

The philosophy underlying the techniques developed in this chapter can best be illustrated with an example. Suppose a box is placed in front of you with a number of knobs on one side and several banana sockets on the opposite ends labelled "INPUTS" and "OUTPUTS". Such a box is shown in Figure 2.0.



Variable Parameter Box

Figure 2.0

The box can have an arbitrarily large but finite number of knobs labelled $p_1, p_2, p_3, \ldots, p_j, \ldots, p_m$, and the inputs and outputs are labelled in some suitably distinguishing fashion. The box in Figure 2.0 has twelve knobs, three outputs and two inputs.

The inputs can be stimulated with analogue signals of appropriate amplitude and the outputs signals measured to some prescribed accuracy. The knobs are set

to an original position called the good setting but they can be adjusted to any other setting between zero and infinity. It is possible to read the scale associated with each knob to some specified accuracy (say 1%). The original settings may be recorded on a piece of paper.

Diagnosis can be thought of as a game involving the box and an opponent. There are two versions of the game. One is called fault detection. The other more difficult version is called fault isolation. These are played as follows.

The box is turned so the knobs can be viewed only by the opponent. Access to the inputs and outputs is still available to you. Your opponent can manipulate the knobs to any position he likes with the restriction that only one knob may be adjusted during any one turn. At the end of the turn, he must reset the knob to its original position. If you have never had a chance to play with the box, you will have difficulty in knowing if your opponent has changed the position of any knob. But if you supply the box with a signal and observe the output, the output may change when your opponent twists a particular knob. If you are given some preparation time during which you can apply inputs, twist knobs and observe outputs, you might be able to compile a table that relates output changes to knob changes for certain inputs. With reasonable preparation time, you may be able to develop a table similar to that shown in Figure 2.1.

The table in Figure 2.1 shows the input number, the input signal applied, the output signals when the knobs are in the good position and lists the knobs which make the output value change from the good value. In general, the BAD column will contain many different values and

these are not listed. The hypothetical table shows that four inputs must be applied during each turn to determine if any knob has been set to a position different from the good position. Once a player has developed his table, he is likely to lose interest in the game because he has a "formula" for determining which knob has been turned. Better players may develop shorter tables but the conclusions reached will be the same

INPUT		OUTPUT			
No.	Signal	No.	Good	Bad	Knobs Effecting Output Change
1	10sint	2	5.5sint		p_1, p_5
2	10sint	3	3sint		p_1, p_2, p_3
1	1	2	10.5		$p_5, p_7, p_8, p_{10}, p_{11}$
1	10	1 3	$5e^{-t}$ 6		p_4, p_6, p_7 p_{12}, p_9, p_3

Fault Detection Table

Figure 2.1

The alternative version of the game is much more interesting and more complex. Developing a method for selecting <u>which</u> knob has been turned, and how much, is in general more difficult than for detecting that some knob has been changed. Without making some assumptions, the game is probably impossible.

Assumptions which permit a playable game to be developed are the following:

1. Knobs can be changed one-at-a-time only.

2. The knobs cannot be continuously varied. Only certain discrete settings are allowed.

3. Certain of the knobs have a stronger influence on a particular output than do the others.

The first assumption limits the decision space to a tractable size. The second assumption is important because it ensures that the set of possible output values is finite. The smaller the step size, the larger the number of elements in the output set. The third assumption may be regarded as a statement on the orthogonality of the outputs.

The strategy for preparing a table to use in the second version is somewhat similar to that used in preparing the first table, (Figure 2.1). It is apparent however, that a table for fault isolation must contain much more information than for fault detection. If the precise setting of the knob is desired, there must be a separate entry for each parameter and another for each setting (see the second assumption above). The development of a fault table can be very time consuming even if the assumptions above apply.

The box game is nearly anlogous to the analogue network diagnosis problem but enough different to require qualification. First, the assumption that parameters change one-at-a-time only is not satisfying from the practical point of view. Parameter drift will be influenced by ageing, temperature and other environmental factors. Theoretically, the problem of directly computing the actual parameter values can be handled analytically only in certain cases.[13] If, for example, the network can be modelled by a set of linear algebraic equations, then the parameter settings

may be any of the combinations permissible in a real network. For given settings of the parameters, a network solution can be computed by analytical techniques. Using an extensive number of solutions and table-look-up techniques, it is possible to select a set of parameters which gives input-output signals close to those obtained. However, this method is lengthy, incomplete and has uniqueness problems.

Even though component parameters drift in the continuum, we postulate that the physical process can be modelled, for fault detection tests, by small discrete parameter changes. Furthermore, in the physical network, there will be a first parameter to drift to a value which causes the output to deviate from its good value. Therefore, if tests are applied often enough, the first parameter to drift will be detected. In Sections 2.7 and 2.8 we extend this idea to develop fault isolation tests.

To recapitulate, in the box game, inputs may be associated with network inputs and the box outputs may be associated with network outputs. We must know the initial state to be able to obtain meaningful response information from a real network. The box reverts to an initial state on the resetting of each knob. Knob settings are analogous to parameter values in a network model.

Physical components subject to ageing will drift on the continuum between some limits. If the possibility of catastrophic failures is neglected, then the parameter values will lie within drift limits which depend on the type and manufacture of the component. For computational purposes, we will assume that the parameters change in very small increments. In theory

the number of increments may be as large as desired, consistent with the accuracy of the computing machine that is used. We will show in Section 2.8 that it is possible to apply discrete computation methods and still regard a parameter value as continuously variable over an interval.

2.2 MODELLING ANALOGUE NETWORKS

The basic network description given in Chapter 1 is in terms of the mapping operation. Because a network can have memory, a single mapping is often insufficient. The set theoretic model described in Equation 1.15, Chapter 1, will be taken as the basic model. In cases, where the network contains no memory or is operating in the steady state, the next state mapping, σ, will not be required because the mapping is either a constant or the co-domain is the null set. For the general case, the type of coupling that occurs in the network must be investigated to determine whether the "next state" mapping will be required.

In this section, the important analogue network models are reviewed. The three classes developed are: d.c., a.c. and time domain models. We then proceed to demonstrate the parameter value solvability and sensitivity measure ideas using the first two. Although the resistive and steady state models are not sufficiently general for the purposes required here, they are useful to illustrate particular concepts.

The general network model presented in Chapter 1 is particularised in this chapter by expressing the structure-parameter value-physical law relationships in algebraic form.

2.2.0 Models and Mappings**

In reducing the general set theoretic model or in modifying it to describe a particular type of analogue network, the relationship between the inputs, outputs and parameters in terms of the type of mapping should be specified. We define

$$F : \underline{u} \rightarrow \underline{y}$$

to mean that for all values of the time, the image of $\underline{u}$ is $\underline{y}$ according to the rule F.*

To describe a network model with no memory, the mapping can be made explicit as the algebraic equation

$$\underline{A}_1 \underline{u} = \underline{y}$$

where $\underline{u}$ is the input vector, $\underline{y}$ is the output vector and $\underline{A}$ is a matrix.

If the mapping is dependent on a parameter α and $\underline{A}$ is a transformation matrix describing the mapping, we can write

$$\underline{A}(\alpha)\underline{u}(\alpha) = \underline{y}(\alpha)$$

where the input, output and $\underline{A}$ are all functions of the variable α. In other words, for a given value of α, say α_1, the mapping from $\underline{u}$ to $\underline{y}$ will be given by a particular rule which will be different from the rule that assign $\underline{y}$ to $\underline{u}$ for another value of α. This idea is analogous to the parameter dependent mapping $M|p_i$ defined in Chapter 1.

---

* The line under a lower case symbol means a vector. The line under an upper case symbol denotes a matrix. (Bold face symbols are difficult to obtain without changing typewriter ribbon.)

** It is recommended that the reader, familiar with basic network and control theory, skip to Section 2.2.5.

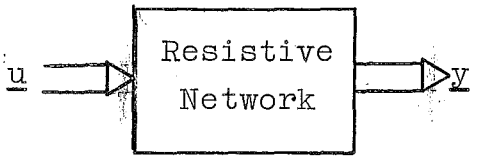
These ideas can be sharpened by looking at the most important subclasses of analogue network models. Those of practical importance are:

1. Resistive - memoryless, dc networks.
2. Reactive - ac sinusoidal steady state networks.
3. Transient networks - total solution.
   (a) Frequency domain models.
   (b) Time domain models.

Special techniques are available to solve each of the above three models. The first two correspond to special cases of the particular integral solution in differential equation theory. Number 3 above considers the total solution including initial conditions, the natural response and any steady state solution. The three cases are outlined to amplify the differences in their mathematical models.

1. Resistive Networks

The basic system diagram for a resistive network is



where $\underline{u}$ is the vector of inputs*, $\underline{y}$ is the vector of outputs.

Basic solution equation:

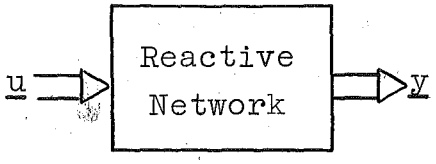
$$\underline{y} = \underline{A}\,\underline{u}$$

*Note that we are departing slightly from previous notation and are calling the input vector $\underline{u}$ rather than $\underline{x}$. In most texts on state variable methods, the symbol $\underline{x}$ is reserved for the state. Brackets, [.] denote a matrix.

where $\underline{A}$ is the $n \times n$ solution matrix of real elements and $\underline{u}$ and $\underline{y}$ may be $n \times 1$ column vectors or scalar functions of time.

2. Reactive ac Networks

The block diagram for reactive networks is basically the same as for resistive networks.



$\underline{u}$ is the vector of inputs, $\underline{y}$ is the vector of outputs.

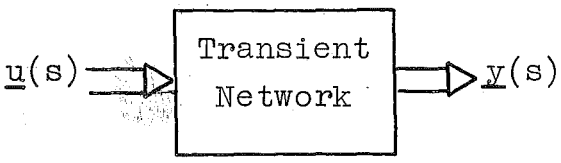
Basic solution equation:

$$\underline{y}(j\omega) = \underline{A}(j\omega) \quad \underline{u}(j\omega)$$

where $\underline{A}(j\omega)$ is an $n \times n$ solution matrix of complex elements and $\underline{u}$ and $\underline{y}$ are in general complex and functions of the frequency variable ω and $n \times 1$.

3. Transient Networks*

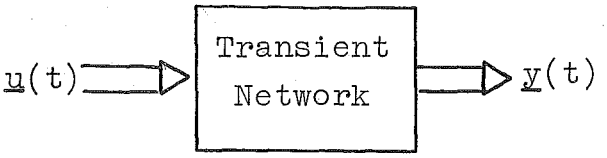
The system diagram for transient networks can be expressed in several forms. First there is the frequency domain diagram



* By transient we mean the set of all possible time domain responses including initial condition solutions.

or the time domain diagram

(b)



where $\underline{u}(s)$ and $\underline{u}(t)$ are input vectors, and
$\underline{y}(s)$ and $\underline{y}(t)$ are output vectors.

The solution methods for the time domain model will now be discussed in some detail.

2.2.1 The General Network Model

The basic solution equations for the s domain can be put into a form similar to the dc and ac cases. However, the time domain formulation requires more explanation.

The general differential equation model of a network is given by

$$\sum_{j=1}^{n} b_{ij} y_i^{j-1}(t) = \sum_{k=1}^{m} a_{ik} u_k^{k-1}(t) \qquad (2.1)$$

where a_{ik} and b_{ij} are coefficients of the inputs and outputs respectively; these also have interpretation as the network zeros and poles. The notation y_i^{j-1} and u_k^{k-1} stands for the $(j-1)$th and $(k-1)$th derivative of y_i of u_k respectively. In general, if the determinant of $[b_{ij}]$, $|b_{ij}|$, is of rank n, y_n outputs can be found. There is a set of integral equations, equivalent to Equation 2.1, which will not be discussed here. It should be noted, however, that numerical techniques for

integration are susceptable to various types of errors. Because of the problems encountered with numerical integration techniques for solving differential equations, the state variable formulation will be preferred. The reason for this preference is that errors in matrix manipulations are generally easier to pin down.[114]

2.2.2 Frequency Domain Model

The solution to Equation 2.1 is often rather difficult to compute by hand. If the original equations have been differentiated to obtain Equation 2.1, many of the initial conditions may be "lost". It is then necessary to apply the boundary conditions to obtain the correct solution. Alternatively, the original intergrodifferential equations can be Laplace transformed. This results in an algebraic model - which can be manipulated to solve for the unknowns. The expression for the frequency domain solution is

$$\underline{y}(s) = \underline{H}(s)\underline{u}(s) \qquad (2.2)$$

where $\underline{y}$ is the vector of outputs, $\underline{H}$ is the system function, and $\underline{u}$ is the input vector.

The solution given by Equation 2.2 does not explicitly include initial condition terms. If the particular initial conditions are included in the $\underline{u}$ vector, this form can be used to find a total solution.

2.2.3 Time Domain Model: Classical Formulation

Linearity is a useful abstraction. The theoretician delights in solutions to linear problems; the pragmatist suffers the non-linear. The suffering is minimized if non-linearities can be separated from the linear parts. We are going to assume that non-linear network models

which are to be tested will be available in solved form.

It is well known that the response of a linear network with no input driving function is obtained as the network homogeneous differential equation solution.[114] A time function, h(t), can be obtained as the δ function response or by cross-correlating the network output with the input when the driving function is white noise. This response function is termed the system response function, h(t). The total output solution including both initial conditions and the forcing function can be found by convolving the system function with the driving function and adding the appropriately weighted natural solution. The total output response can then be expressed as

$$\underline{y}(t) = \underline{h}'(t,t_o) + \int_{t_o}^{t} \underline{h}(t-\tau)\underline{u}(\tau)d\tau \qquad (2.3)$$

The second term on the right in Equation 2.3 is the convolution integral and it is usually written as $\underline{h}(t) * \underline{u}(t)$ for brevity. $\underline{h}'(t,t_o)$ is the weighted natural solution. It can be shown[114], that there is a function $\underline{H}(s)$, equivalent to h(t) in Equation 2.2 which is the Laplace transform of $\underline{h}(t)$ if the initial conditions are zero. That is,

$$\mathcal{L}[\underline{h}(t)] = \underline{H}(s)$$

and $\mathcal{L}$ is the symbol for the operation which transforms h(t) to the s domain.

2.2.4 Time Domain Analogue Network Models: State Variable Method

State variable methods proffer the most systematic method available for obtaining solutions for linear networks in the time domain. They express the total

solution to the network directly using matrix methods.

The revival of these methods, for example see Ince[62] - sometimes called state space methods - is partially due to the advent of digital computers which render solutions to matrix equations resulting from the state variable expressions. Credit should also be given to Kalman and Bucy[68] and to Zadeh and Desoer[148] for giving certain impetus and inspiration to the revival. They are of interest here because they can be programmed for digital computer solution[76].

State space methods are not of interest only because they offer systematic solutions; equally important are the ideas of observability and identification which follow naturally from this formulation. These two ideas are important because of the close relationship they bear to fault detection and isolation. This relationship will be discussed briefly in Section 2.3.

Equation 1.3 in Chapter 1 describes a process which has an output that is affected by its present environment and its past history. The environment enters the process through the inputs and the history is recorded in the state. Under certain conditions, it is necessary to know only the present environment and a portion of the history to know how the process will proceed. If the laws governing the process are simple, (linear for example) mappings can be developed which describe how the system proceeds with time.

A correspondence between the general set theoretic network model of Equation 1.3 Chapter 1 and the state variable model exists if the system function is regarded as a mapping of some present state into the next state. This is precisely what happens when $\underline{u}$ is zero. Then a

composite mapping involving the state and input vectors which depends on the superposition property, is used if the total state response is required. (See for precise development, Zadeh and Desoer, Chapters 1 and 2; also the points made by Arbib, and Zeiger[3] are relevant.)

If a network is to be diagnosed, a state response which excites each mode must be obtained. In addition every state must be observable.

The state space network equations describing a linear differential network are

$$\dot{\underline{x}} = \underline{A}(t)\underline{x} + \underline{B}(t)\underline{u} \quad (a)$$
$$\underline{y} = \underline{C}(t)\underline{x} + \underline{D}(t)\underline{u} \quad (b) \qquad (2.4)$$

where $\underline{x}$ is the column state vector and $\underline{u}$ is the column control or driving function and $\underline{y}$ is the output; $\underline{A}(t)$, $\underline{B}(t)$, $\underline{C}(t)$ and $\underline{D}(t)$ are matrices and as shown, may be time varying. $\underline{x}$, $\underline{y}$ and $\underline{u}$ may be vectors or scalars. For the latter condition, matrices will be regarded as vectors or coefficients. A diagram of the equations is shown in Figure 2.2.

Diagram of State Equation Relationships

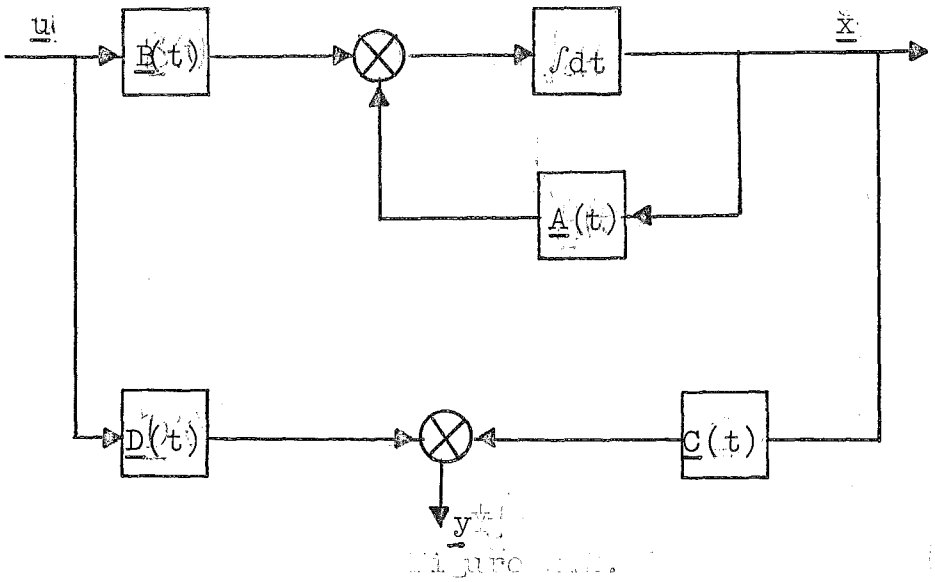


Figure 2.2

It has been shown[76] that the state solutions to the network differential equation (Equation 2.1) for time varying parameters can be expressed in terms of the state transition matrix Φ as

$$\underline{x}(t) = \underline{\Phi}(t,t_o)\underline{x}_o + \int_{t_o}^{t} \underline{\Phi}(t,\tau)\underline{B}(\tau)\underline{u}(\tau)d\tau \qquad (2.5)$$

where $\underline{\Phi}(t,t_o) = e^{\int_{t_o}^{t} A(\tau)d\tau}$ and other symbols have previous meanings. x_o is the initial state.

The first term in Equation 2.5 corresponds to the classical transient solution and the second term is the particular integral or forced solution. {Note the difference between Equations 2.5 and 2.3; the solution is in terms of the state vector $\underline{x}$ and not the output vector $\underline{y}$.} In the absence of a forcing function (control) the second term is zero. The <u>adjoint equation</u>[148] is used to compute $\underline{\Phi}(t,\tau)$ for various τ. It is given by

$$\frac{d}{d\tau}\underline{\Phi}^T(t,\tau) = -A^T(\tau)\underline{\Phi}^T(t,\tau), \quad \underline{\Phi}(t_o,t_o) = \mathcal{U}$$

for the fixed t. Superscript T means transpose and $\mathcal{U}$ is the identity matrix.

There are two drawbacks to using Equation 2.5 for computing diagnostic information. The first is that the identity of individual elements in the A matrix is lost through transformations. Second, the state variables are not directly measurable in most cases. Consequently, the resulting expression is not particularly useful for diagnostic studies.

A better form for parameter variation computations retains the basic differential equation solution matrices.

In this case Equation 2.5 becomes

$$\underline{x}(t) = \underline{I}e^{\underline{A}(t-t_o)}\underline{x}_o + \int_{t_o}^{t} \underline{I}e^{\underline{A}(t-\tau)}\underline{B}(\tau)\,\underline{u}(\tau)\,d\tau \quad (2.6)^*$$

where the network parameters a_{ij} and b_{ij} should be expressed in literal form. If the system is time invariant, the matrix elements are constants made up of combinations of network piece-part parameters.

For the time invariant case, Equation 2.3 becomes

$$x(t) = e^{\underline{A}(t-t_o)}\underline{x}_o + \int_{t_o}^{t} e^{\underline{A}(t-\tau)}\underline{B}\,\underline{u}(\tau)d\tau, \quad (2.7)$$

Finally, the observable output vector is given by substituting 2.7 into 2.4(b), which gives

$$\underline{y}(t) = \underline{C}\,e^{\underline{A}t}\underline{x}_o + \int_{o}^{t} \underline{C}\,e^{\underline{A}(t-\tau)}\underline{B}\,\underline{u}(\tau)d\tau + \underline{D}\,\underline{u} \quad (2.8)$$

where t_o is taken to be zero.

2.2.5 Conditions for Fault Detection and Isolation

The functional expression given explicitly by Equation 2.8 for the output $\underline{y}$ in terms of the input vector $\underline{u}$, the network state parameters a_{ij}, the output network driving function parameters b_{kl}, the output observability parameters c_{pq}, transmission parameters d_{rs} and time can be written implicitly as

$$\underline{y} = \underline{G}(\underline{u}, a_{ij}, b_{kl}, c_{pq}, t) + \underline{\Gamma}(d_{rs}, \underline{u}, t) .$$

---

* $Ie^{\underline{A}(t-t_o)}$ stands for $e^{\int_{t_o}^{t}\underline{A}(\tau)d\tau}$

It will be assumed that the a, b, c and d parameters are time invariant. The functions $\underline{G}$ and $\underline{\Gamma}$ can be identified with components in Equation 2.8.

For fault detection, changes in the output which result from changes in network piece part values are of interest. The expression

$$\frac{\partial \underline{y}}{\partial a_{ij}} \tag{2.9}$$

is a measure of the influence that a parameter a_{ij} in the $\underline{A}$ matrix can have on the observable outputs. Because the $\underline{A}$ matrix contains information not only on memory elements but also on the memoryless elements, it is possible to obtain a measure of the effect of a variation in the network parameter p_k, on the a_{ij} elements by computing

$$\frac{\partial a_{ij}}{\partial p_k} \tag{2.10}$$

for all branch parameters p_k in the network.

A necessary condition for developing fault detection tests corresponding to changes in p_k is that

$$\frac{\partial a_{ij}}{\partial p_k} \neq 0$$

for at least one a_{ij}, and all p_k in the network.

A necessary and sufficient condition for fault detection in a linear differential network is, from Equation 2.9 and 2.10,

$$\frac{\partial y(t_\alpha)}{\partial p_k} \neq 0 \tag{2.11}$$

for all p_k and at least one y_j and for at least one

value of time, t_α. No proof has been developed here but basically, if a change in p_k gives rise to the presence of first order terms in the Taylors expansion of the output trajectory y_j around the point t_α, the trajectory with p_k will be different from the trajectory with $p_k + \Delta p_k$ substituted. Δp_k is a small change in p_k.

2.3 OBSERVABILITY, IDENTIFICATION AND DIAGNOSIS

If the matrix $\underline{C}$ in Equation 2.8 has the kth column all zeros, the state x_k is not reflected in the output vector. x_k is not observable. If the matrix $\underline{C}$ has rank equal to the $\underline{A}$ matrix, then the modes of the network are observable. (148) Because all network piece-part parameters will appear in one or more of the a_{ij}, it is reasonable to assume that the change in value of a piece-part will affect several outputs.

Identification involves the problem of estimating the values of the a_{ij} from input-output measurements. (68) The a_{ij} are initially unknown and generally, the structure or the order of the network is guessed.

Analogue network diagnosis is concerned with determining if an initially known a_{ij} has changed - the detection problem - and if a change has occurred, which piece-part parameter p_k, is responsible for the changes in the a_{ij}'s - the isolation problem.

2.4 DISCUSSION

All subsequent discussion assumes that a network model which describes the behaviour of the system in the time domain is available. A temporary exception to this

occurs in the next section on output sensitivity to parameter changes and again in Section 2.10 on transfer function methods. We use an example in Section 2.5 to illustrate the difficulty in obtaining parameter variation information directly from the mathematical model. Because of the excessive computation required for analytically solving a simple d.c. network case, the assertion is made that simulation of parameter variations using the network model in conjunction with a digital computer program offers a more tractable approach.

If we are interested in fault detection only, we simulate the network and obtain the output sensitivity with a computing algorithm instead of using the analytical approach described the next section. The output sensitivity can be used to select outputs to measure for fault detection. Knowing what the good outputs should be, a difference Δy can be used to detect parameter changes.

In Section 2.10 transfer function diagnostic techniques are discussed. Methods for indirectly estimating and directly computing network parameter values by generating equations using input-output information sufficient for solving for the parameters directly are mentioned. However, these methods are not viable in many system environments in which on-line techniques are required. Even more objectionable is the fact that large matrices must be inverted in the process of computing parameter values.

The philosophy underlying the development in the remainder of this chapter is founded on one assertion. It can be stated as follows:

Assertion: If the time domain sensitivity of a

network output defined by

$$\frac{\partial y_j(\hat{P},\underline{u},t)}{\partial p_i} \qquad (2.12)$$

is non zero, where y_j is a particular output vector component, $\hat{P}$ is the design parameter set, p_i is a piece-part parameter, $\underline{u}$ is the input and t is time, it is possible to use the output y_j to develop diagnostic information for the network with respect of changes in the value of the parameter, p_i.

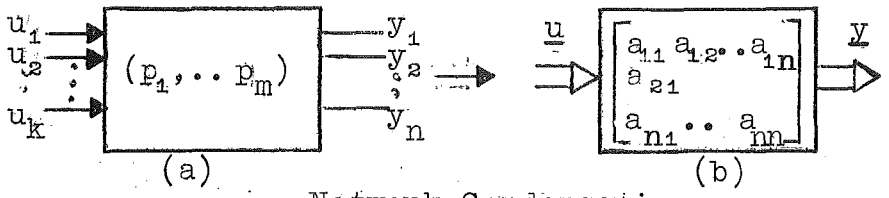
This means that if the design parameters $\hat{P}$ (called the good set), givena particular output trajectory, then a deviation in parameter p_i will change the output trajectory, assuming that the input is fixed, the structure is fixed, and the initial state is fixed. In practice, it may be difficult to fix the initial state. However, if the output is computed for a variety of different initial state vectors, the effect on the other than nominal vector on the output can be evaluated. This problem bears resemblance to the sequential machine identification problem (107) and investigations to determine the effect that different initial states have on the output should be made. Zadeh and Desoer[148] have shown that by appropriate selection of the initial state vector, a single mode of the network can be excited.* This result illustrates a property that could be potentially useful for diagnosis. Moreover it points to possibly spurious conclusions if test algorithms do not specify a fixed set of initial conditions.

---

* Here, of course, the network is a linear differential network.

2.5 FAULT DETECTION USING OUTPUT SENSITIVITIES

The basic mathematical model for the d.c network described in Section 2.2.0 results in a "condensation" of the parameter value information. The process is depicted in Figure 2.3 where we assume that parameters p_i are piece-part resistance values, $\underline{u}$'s are independent voltages and $\underline{y}$'s are dependent currents.



Network Condensation

Figure 2.3

In going from (a) to (b) in Figure 2.3, the actual parameter values p_i combine to form the a_{ij} parameters through what may be regarded as a coding process. The coding process is the combining - by addition, subtraction, multiplication and division - of the actual parameter values $\{p_1, p_2, \ldots, p_m\}$ into the set of matrix elements, a_{ij}. We can write this as

$$a_{ij} = a_{ij}(p_1, p_2, \ldots, p_m) \tag{2.13}$$

where the matrix of a_{ij}'s is $n \times n$. Suppose that it is desired to solve for the currents. Starting with the equation

$$\underline{A}\,\underline{y} = \underline{u} \tag{2.14}$$

we premultiply by the inverse of $\underline{A}$ and obtain

$$\underline{y} = \underline{B}\,\underline{u} \tag{2.15}$$

where $\underline{B} = \underline{A}^{-1}$. Now by definition, the inverse of $\underline{A}$ is

$$\underline{A}^{-1} = \frac{\text{adj}\,\underline{A}}{/\underline{A}/} \tag{2.16}$$

where adj$\underline{A}$ is the adjoint of $\underline{A}$ and /A/ is the determinant of $\underline{A}$. Because /$\underline{A}$/ depends on every element of the $\underline{A}$ matrix and terms of the adjoint matrix (the cofactors of $\underline{A}$), the currents $\underline{y}$ and each term of the $\underline{B}$ matrix, b_{ij}, will be some function of all of the a_{ij}'s. This can be written as

$$b_{ij} = b_{ij}(a_{11}, a_{12}, \ldots a_{ij}, \ldots, a_{nn}) \qquad 2.17$$

for $\underline{A}$ and $\underline{B}$, $n \times n$. Combining Equations 2.13 and 2.17 we obtain

$$b_{ij} = b_{ij}(a_{11}(p'_m), a_{12}(p'_m) \ldots, a_{nn}(p'_m)) \qquad 2.18$$

where p'_m denotes the set $\{p_1, p_2, p_3, \ldots, p_m\}$.

2.5.0 Output Sensitivity

The output sensitivity to a parameter change can be developed by selecting a single output to work with. Thinking in terms of equipment and functional element definitions from Chapter 1, we can regard the selected output as the measurable equipment output and the parameters as functional element parameters. Or, from the functional element point of view, the output would be dependent on the PENET parameters - in this case resistance values. Selecting a particular output, y_k, and using Equation 2.15 we obtain

$$y_k = (b_{k1}, b_{k2}, \ldots, b_{kn}) \cdot (u_1 u_2, \ldots u_n)^T \qquad 2.19$$

where the solution for y_k is the kth row of $\underline{B}$ multiplied by the transpose of $\underline{u}$. The linear variation of y_k with

respect to the b_{kj}'s is defined as

$$dy_k = \frac{\partial y_k}{\partial b_{k1}} db_{k1} + \frac{\partial y_k}{\partial b_{k2}} db_{k2} + \ldots + \frac{\partial y_k}{\partial b_{kn}} db_{kn} \qquad 2.20$$

where partial derivatives are with respect to all b_{kj} for k fixed. From Equations 2.13 and 2.18 we compute the db_{kj} and da_{ij} terms as

$$db_{kj} = \frac{\partial b_{kj}}{\partial a_{11}} da_{11} + \frac{\partial b_{kj}}{\partial a_{12}} da_{12} + \ldots + \frac{\partial b_{kj}}{\partial a_{nn}} da_{nn} \qquad 2.21$$

and

$$da = \frac{\partial a_{11}}{\partial p_1} dp_1 + \frac{\partial a_{11}}{\partial p_2} dp_2 + \ldots + \frac{\partial a_{11}}{\partial p_m} dp_m \qquad 2.22$$

and similarly for $a_{12}, a_{13}, \ldots, a_{nn}$. Substituting 2.21 and 2.22 into 2.20 we obtain the linear variation in y_k as

$$dy_k = \sum_{j=1}^{n} \sum_{q=1}^{n} \sum_{l=1}^{n} \frac{\partial y_k}{\partial b_{kj}} \frac{\partial b_{kj}}{\partial a_{ql}} \left(\frac{\partial a_{ql}}{\partial p_1} dp_1 + \frac{\partial a_{ql}}{\partial p_2} dp_2 + \ldots + \frac{\partial a_{ql}}{\partial p_m} \right) dp_m \qquad 2.23$$

The terms of interest in Equation 2.23 are those which give the linear variation in y_k with respect to each p_i.

If the a_{ij} were directly measurable, their first derivatives with respect to each p_q could be used in Equation 2.23. For diagnosis, the outputs are the only measurable quantities. Therefore, for fault studies we require to know how much information about changes in

a parameter, p_i can be got by measuring all y's. Because the form of Equation 2.23 does not suggest a method for practical measurements, we will assume that it is possible to compute the components of the linear variation dy_k for one-at-a-time variations in the parameters themselves. The quantity to be computed will be defined by

$$\mathcal{S}\, y_k \Big|_{p_i} = \frac{\partial y_k}{\partial p_i} \qquad 2.24$$

and will be called "the sensitivity of y_k with respect to parameter i".

Equation 2.24 is closely related to the classical sensitivity measure [35] often used for stability analysis. However, the application in this Section will be for computing fault detection measures rather than for determining the effect of parameter changes on stability. A deviation in the output y_k from its mean value, $y_k(\text{mean})$, can be computed for changes in all the p_i. The values obtained may lie outside the envelope that defines the allowable output trajectory range. (See Section 1.8, Chapter 1)

If only degradation failures are assumed, a maximum positive and negative deviation in y_k for one-at-a-time parameter variations can be found. The actual value of y_k is computed by setting each p_i to the "drift" tolerance level and solving the network equations for y_k in each case. A fault corresponds to a y_k value outside the envelope computed in the previous step. Breen and Webb have reported a scheme similar to this used for developing fault dictionaries which give a table of computed y_k verses $p_i \pm \Delta p_i$ where the p_i are varied one-at-a-time. [142] The computation is done in

the matrix equation describing the network. It gives an "envelope" of output values which the technician trouble-shooting the network can use to compare with actual measured test point values. He selects the set of outputs from the table which most closely approximate the measured values. By noting which parameter(s) caused the present values, he is able to narrow down the possible faulty components to a few. Breen and Webb use the human operator to interpret the data to obtain fault isolation information.

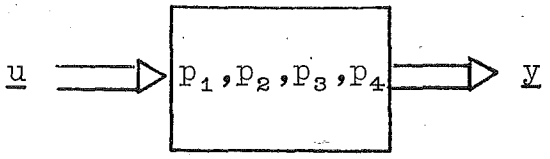
The analytical problem of computing Equation 2.24 can be circumvented by using a parameter variation simulation method. This is done by writing the network model in matrix form or in the solved form and subsequently using a computing algorithm to vary the parameters one-at-a-time. The sensitivity can be numerically computed by taking the difference of the solutions for the output with the parameters set to their good values and for the parameters set to their perturbed values (one-at-a-time). From this, numerical values of the sensitivity can be developed.

The sensitivity measure discussed here is useful for fault detection but, as mentioned previously, not directly applicable for fault isolation. However, by using a sequential decision process (viz a test diagram or fault table) it is possible to narrow the set of faults and in some cases obtain isolation.

2.5.1 Direct Computation of Sensitivities: Example

The technique of analytically computing the sensitivity for an algebraic network model (the dc case

can be illustrated by using the model shown in Figure 2.4.



Four Parameter Network

Figure 2.4

Suppose that the network equations have been written and that the parameters are related to the a-parameters by

$$a_{11} = p_1 p_2$$
$$a_{21} = p_1$$
$$a_{12} = p_3 p_2 p_4$$
$$a_{22} = p_4 p_1 \quad .$$

We can write the two port network equations in matrix form as

$$\underline{u} = \underline{A}\,\underline{y} \qquad 2.25$$

where $\underline{u} = (u_1 u_2)^T$, $y = (y_1 y_2)^T$, $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$

Solving Equation 2.25 we find

$$y = \underline{A}^{-1}\, u \qquad 2.26$$
$$= \underline{B}\,\underline{u}$$

where $$B = \begin{vmatrix} \frac{a_{22}}{\Delta_A} & \frac{-a_{21}}{\Delta_A} \\ \frac{-a_{12}}{\Delta_A} & \frac{a_{11}}{\Delta_A} \end{vmatrix} = \begin{vmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{vmatrix}$$

and $$\Delta_A = /A/ = a_{11}a_{22} - a_{21}a_{12} \quad .$$

The linear variations in the outputs dy_1 and dy_2 are given by

$$dy_1 = \frac{\partial y_1}{\partial u_1} du_1 + \frac{\partial y_1}{\partial u_2} du_2 + \frac{\partial y_1}{\partial b_{11}} db_{11} + \frac{\partial y_1}{\partial b_{12}} db_{12} + \frac{\partial y_1}{\partial b_{21}} db_{21} + \frac{\partial y_1}{\partial b_{22}} db_{22} \qquad 2.27$$

and

$$dy_2 = \frac{\partial y_2}{\partial u_1} du_1 + \frac{\partial y_2}{\partial u_2} du_2 + \frac{\partial y_2}{\partial b_{11}} db_{11} + \frac{\partial y_2}{\partial b_{12}} db_{12} + \frac{\partial y_2}{\partial b_{21}} db_{21} + \frac{\partial y_2}{\partial b_{22}} db_{22} .$$

If the input voltages, u_1 and u_2 are assumed constant, then a typical term in Equation 2.27 will be

$$\frac{\partial y_k}{\partial b_{ij}} db_{ij} \quad . \qquad 2.28$$

The expression given in Equation 2.28 contains the parameter information "doubly transformed". It was shown in Equation 2.21 that the term db_{ij} can be written

$$db_{ij} = \frac{\partial b_{ij}}{\partial a_{kl}} da_{kl} \qquad (a) \qquad 2.29$$

and by Equation 2.22,

$$da_{kl} = \frac{\partial a_{kl}}{\partial p_q} dp_q \quad (b) \; . \qquad 2.29$$

Substituting Equation 2.29 (a) and (b) into Equation 2.28 gives

$$\frac{\partial y_k}{\partial b_{ij}} \frac{\partial b_{ij}}{\partial a_{kl}} \frac{\partial a_{kl}}{\partial p_q} dp_q = \frac{\partial y_k}{\partial p_q} dp_q \qquad 2.30$$

This expresses the variation in y_k directly as a function of changes in the p_q. For our 2×2 examples, terms on the left of Equation 2.30 are computed in Table 2.0.

Computation for the first term of Equation 2.30 is completed by finding the partial derivative of y_k with respect to each b_{ij}. The dp_q terms may be treated as the actual component parameter deviations due to drift. Table 2.0 (a) possesses considerable symmetry and the triangle including the main diagonal is all that need be computed. The other terms can be filled in by noting that

$$\frac{\partial b_{ij}}{\partial a_{kl}} = \frac{\partial b_{ij}}{\partial a_{lk}} \; .$$

The p_q can be substituted in the a_{ij} terms and all of the entries in Table 2.0 (a) and (b) will be in terms of the p_q. Finally, an expression for $\delta y_{k|p_i}$ (after Equation 2.24) can be computed. For example, the sensitivity of y_1 with respect to p_1 is, using Equation 2.23.

$\frac{\partial b_{ij}}{\partial a_{kl}}$ (a)

	a_{11}	a_{12}	a_{21}	a_{22}
b_{11}	$\frac{-a_{22}^2}{\Delta_A^2}$	$\frac{a_{21}a_{22}}{\Delta_A^2}$	$\frac{a_{12}a_{22}}{\Delta_A^2}$	$\frac{\Delta_A - a_{11}a_{22}}{\Delta_A^2}$
b_{12}	$\frac{a_{21}a_{22}}{\Delta_A^2}$	$\frac{-a_{21}^2}{\Delta_A^2}$	$\frac{-\Delta_A - a_{21}a_{12}}{\Delta_A^2}$	$\frac{+a_{21}a_{11}}{\Delta_A^2}$
b_{21}	$\frac{+a_{12}a_{22}}{\Delta_A^2}$	$\frac{-\Delta_A - a_{12}a_{21}}{\Delta_A^2}$	$\frac{-a_{12}^2}{\Delta_A^2}$	$\frac{a_{12}a_{11}}{\Delta_A^2}$
b_{22}	$\frac{\Delta_A - a_{11}a_{22}}{\Delta_A^2}$	$\frac{a_{21}a_{11}}{\Delta_A^2}$	$\frac{a_{12}a_{11}}{\Delta_A^2}$	$\frac{-a_{11}^2}{\Delta_A^2}$

$\frac{\partial a_{kl}}{\partial p_q}$ (b

	p_1	p_2	p_3	p_4
a_{11}	p_2	p_1	0	0
a_{12}	0	p_3p_4	p_2p_4	p_2p_3
a_{21}	1	0	0	0
a_{22}	$\frac{-p_4}{p_1^2}$	0	0	$\frac{1}{p_1}$

Partial Derivative Table

Table 2.0

$$\delta y_1\Big|_{p_1} = u_1\left(\frac{-a_{22}^2}{\Delta_A^2}p_2 + \frac{a_{12}a_{22}}{\Delta_A^2} + \left(\frac{\Delta_A - a_{11}a_{22}}{\Delta_A^2}\right)\left(-\frac{p_4}{p_1^2}\right)\right)$$

$$+ u_2\left(\frac{a_{21}a_{22}}{\Delta_A^2}p_2 + \frac{-\Delta_A^2 - a_{12}a_{21}}{\Delta_A^2} + \left(\frac{a_{21}a_{11}}{\Delta_A^2}\right)\left(-\frac{p_4}{p_1^2}\right)\right)$$

$$+ u_1\left(\frac{a_{12}a_{22}}{\Delta_A^2}p_2 + \frac{-a_{12}^2}{\Delta_A^2} + \frac{a_{12}a_{11}}{\Delta_A^2}\left(-\frac{p_4}{p_1^2}\right)\right)$$

$$+ u_2\left(\frac{\Delta_A - a_{11}a_{22}}{\Delta_A^2}p_2 + \frac{a_{12}a_{11}}{\Delta_A^2} + \frac{a_{11}^2}{\Delta_A^2}\frac{p_4}{p_1}\right)$$

2.31

The a_{ij} in terms of p_i can be substituted. The expression after this substitution will yield an even more complex equation. Nevertheless, the sensitivity of y with respect to each parameter p_i can always be computed using the above technique, when the network model is reduced to its four terminal equivalent. Considerable simplification results if some of the u are zero. This is often the case in practical networks.

2.5.2 Piece Part Parameter Solvability

Once it has been established that a parameter variation effects a change in a certain output, the necessary conditions for performing fault detection are established. If the sensitivity is identically zero, this means that the output signal contains no information, coded or otherwise, about the particular parameter. Consequently, it may be necessary to select several outputs to obtain information sufficient for fault

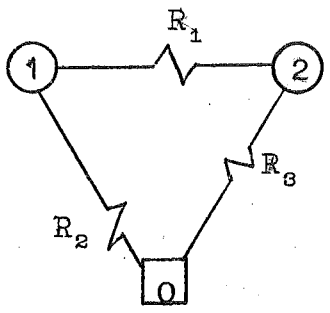
detection.

Berkowitz has shown the conditions for network component parameter value solvability (13). For passive single element kind (resistive) networks, his condition for solvability is

$$N_R \leq \tfrac{1}{2}A(A + 2P - 1) \qquad 2.32$$

where N_R is the number of resistances, A is the number of terminals available for measurement of currents and voltages and for the application of stimuli and P is the number of terminals available for measurement only.

Equation 2.32 can be interpreted as a measure of the number of independent equations which can be generated for a network of given topology and terminal availability. The equations are themselves non-linear but for small networks, the solution can be obtained by substitution. To indicate the process, an example due to Berkowitz is shown in Figure 2.5.



○ - A terminal

□ - P terminal

$N_R = 3,\ A = 2,\ P = 1$

EQUATIONS:

$Y_{11}E_1 = I_1$

$Y_{22}E_1 = I_2$

$Y_{11} = \frac{1}{R_1} + \frac{1}{R_2}\ ;\quad Y_{22} = \frac{1}{R_1} + \frac{1}{R_3}$

$Y_{12} = Y_{21} = \frac{-1}{R_1}$

Berkowitz Example
Figure 2.5

Using the above equations the values for R_1, R_2 and R_3 are

$$R_1 = -\frac{1}{Y_{12}}$$

$$R_2 = \frac{1}{Y_{11} + Y_{12}}$$

$$R_3 = \frac{1}{Y_{22} + Y_{12}}$$

It is assumed that terminal [o] is available for earthing or for measurement and that terminals (1) and (2) can be stimulated and the response measured. Berkowitz has shown even more general conditions than in Equation 2.32 to treat two element kind networks. However, the equations are even more non-linear than the above set and somewhat messy to solve. Nevertheless, this method provides a valuable contribution to the analogue network diagnosis problem because it establishes necessary conditions for being able to directly find the piece part parameter* values assuming the stated restrictions on terminal availability.

Another diagnostic method due to Seshu and Waxman[127] uses steady-state frequency domain methods to detect changes in the frequency response characteristics. They develop a "signature" which can be related back to a particular parameter change. By comparing signatures obtained from physical networks with signatures obtained for the models with various parameter variations introduced, it is possible to predict the location of the faulty component. This technique and a related technique due to Valstar[140] are discussed in detail in Section 2.10.

A recent paper by Weitzenfeld and Happ generalizes

---

* Piece parts are equivalent to PENE defined previously.

Berkowitz's terminal configuration discussion.[143] They generate the x-terminal, n-port non-redundant networks from the z-terminal parent network and their results specify the number of combinations of terminal configurations which can be obtained for a given network.

In the next section we describe algorithms for detecting network parameter changes. The class of techniques developed is intended to be used with time domain models which retain component parameter information in explicit form. These techniques have not, to the best of the author's knowledge, been reported elsewhere.

2.6 TIME DOMAIN FAULT DETECTION AND ISOLATION

2.6.0 Signal Decoding

We have stated that for diagnostic purposes, information on the parameter values must be obtained through input-output measurements. The parameter information is in a coded form and a fault isolation technique must decode the output signals to extract the required information. The signals that appear at the outputs may contain information on a few parameters or all of the parameters and will be contaminated with unwanted noise. We assume that the noise is negligible for the time being.

What we are seeking is a method or a "filter" that extracts or decodes relevant information from the output signals and presents this in a form which may be deciphered and related to the parameter values. If the network is linear, it is possible to identify certain modes (eigenvalues) which characterize the system. The transfer function methods mentioned previously[127] detect

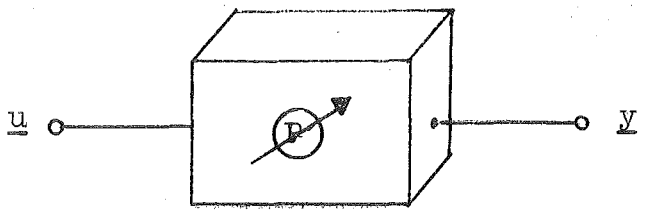
changes in parameter values by detecting shifts in the system function poles. If these poles are reasonably well separated and the network is linear, some success using these methods can be expected.

A basic requirement of the decoding process is the condition that the sensitivity measure given in Equation 2.24 be non zero. The justification for the following methods is largely heuristic; the approach is that of the pragmatist (engineer) who is faced with solving a problem which is not well understood. We postulate a technique and then see if it gives the desired results.

Basically, the input and initial conditions are specified and the good model is solved on a digital computer. The output signal is measured (computed) at discrete time intervals only. The values of the outputs are then quantized to preselected levels. The level within which an output value falls is recorded along with the corresponding time interval. Hence, the good network is characterised by a sequence of numbers which is defined by the quantized output signal, sampled at fixed times throughout a given test measurement period. Next, the network solution is obtained with one of the parameters set to other than the design value. The output sequence under conditions identical to these used to compute the good output is obtained. This process is continued until information sufficient for developing the particular test has been generated. Using information obtained from this general parameter perturbation-simulation method, various types of tests can be defined. As we shall subsequently demonstrate, all methods can be quantatively assessed for their testability measure.

2.6.1 Signal Sequences, Parameter Sets, and Transformations

The system shown in Figure 2.6 represents the fundamental analogue network model. It will be tacitly assumed that its structure is known and invariant. The model simulates faults as model parameter variations. Parameter variations are sensed as mismappings of the input into the output. The variable nature of the parameters is illustrated by the circle with an arrow through it which surrounds the parameter set, P.



Basic Variable Parameter Value Model
Figure 2.6

The general output signal will be denoted by

$$\underline{y} = \underline{y}(\underline{u},P,t) \quad \text{or} \quad \underline{y}(\underline{u},P,t^*)$$

where $\underline{u}$ is the input signal vector, P is the parameter set and t is time. t^* is discrete or sampled time. The good output signals are functions of u, P, and t. From the mapping definition in Chapter 1, we can write

$$M(P,t^*) : P \times \underline{u}(t^*) \rightarrow \underline{y}(t^*).$$

to describe the operation of the network in Figure 2.6,

Because it is convenient to be able to specify the parameter set with one element different, the notation P_r^q will denote the set with the rth parameter at the qth value where

$$r = 1,2,3 \ldots m \text{ and } q = \pm 1, \pm 2, \pm 3 \ldots \pm \frac{k}{2}.$$

Then, the <u>good</u> parameter set will be written

$$P^o \subset P$$

where $\{p_1^o, p_2^o, \ldots, p_m^o\} = P^o$ (a)

and $\{p_1^o, p_2^o, \ldots, p_r^q, \ldots, p_m^o\} = P_r^q$ (b) 2.33

will represent the parameter set with the rth parameter set to q values. Note that the difference between 2.33 (a) and 2.33 (b) is the unlike single element set; that is

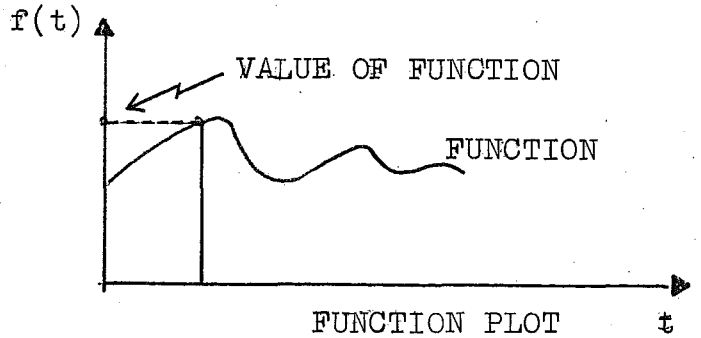
$$P^o - P_r^q = p_r^q \quad .$$

If a particular output signal is denoted by y_j and a particular input signal by u_i, the <u>value</u> of a signal at time t_1 is given by

$y_j(t_1)$ or $y_j(t_1^*)$ - outputs

and $u_i(t_1)$ or $u_i(t_1^*)$ - inputs.

The difference between the function and the value of a function is illustrated in Figure 2.7 for the reader who may be uncertain of the distinction.



FUNCTION PLOT

Figure 2.7

A function may have many different values as shown in Figure 2.7 or it may be a constant function which has the same value for all arguments. The constant function contains no information except perhaps in a very limited sense. On the other hand, a function which has a large set of values for different arguments contains potentially useful information.

The transformation of a function can simplify the representation of the information in a function. It can also make the manipulation of information in the function much easier. Two familiar and very useful transforms are the Fourier Transform and the Laplace Transform. Transformation may be thought of here, as a process which converts the time domain function to an algebraic expression whose argument is $j\omega$ or $s = \sigma + j\omega$. For example, taking the Fourier transform of the output function $y_j(t)$ gives

$$y_j(j\omega) = \mathcal{F}[y_j(t)] = \int_{-\infty}^{\infty} y_j(t)e^{-j\omega t}\,dt \qquad 2.34$$

and the Laplace transform is

$$y_j(s) = \mathcal{L}[y_j(t)] = \int_{-\infty}^{\infty} y_j(t)e^{-st}\,dt \qquad 2.35$$

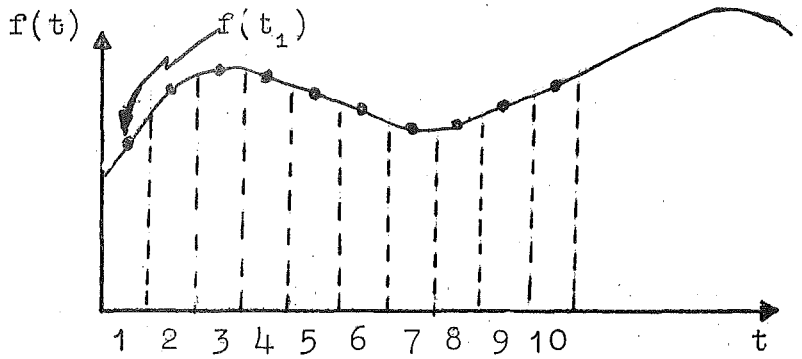
Notice that $y(j\omega)$ and $y(s)$ are frequency domain functions but the argument and domain of the functions are different.

Visualization of the effect of a parameter change is perhaps easier if the model is expressed in the frequency domain. A parameter change will alter the magnitude of certain spectral terms and change their phase. A comparison of all spectral terms is not computationally simple for most functions and is rejected as a method for developing fault isolation techniques because the

interpretation of the spectrum when applied to non-linear networks is difficult and because it has uniqueness problems. However, this approach warrants further investigation because it is potentially applicable to on-line techniques. Where non-linearities are memoryless and can be separated, spectral analysis of the linear network is useful for obtaining diagnostic information.

2.6.2 Tests Based on Binary Relations: Discrete Sampling Method

For fault isolation, we require a measure on the sequence of output values (in the time domain) which will be unique for all possible sequences. Here, all possible sequences means the sequences generated for all possible one-at-a-time faults. If the output signal is sampled and the value of the signal during each interval is associated with an integer, 1,2,3, ..., n, this sequence will be regarded as the sampled signal or the ordered output set as shown in Figure 2.8. If a parameter change is detectable, it will alter the value of one or



Output Signal Set = $\{f(t_1), f(t_2), \ldots, f(t_{10})\}$

Figure 2.8

more elements in the output sequence.* If a single parameter deviation affects the value of at least one element in the output set, the network faults corresponding to the deviation is said to be possibly isolateable. If no other fault condition gives an output sequence having the same range, the network fault is definitely isolateable. We assume here that the output sequence is of necessary extent. (This notion will be clarified in the following paragraphs.)

Suppose that the ranges of the co-domain for two input-output mappings under the same input $\underline{u}$ differs by one element only. This condition is illustrated in Figure 2.9. Obviously, the two mappings M_1 and M_2 are distinct. If we call one output O_1, and the other O_2,

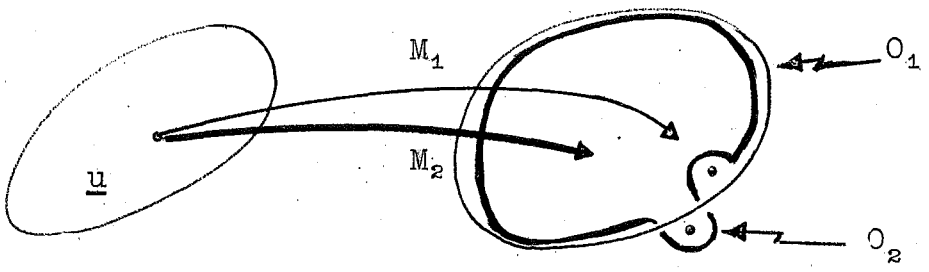


Figure 2.9

where $\underline{u}$ is the set of all input sequences and O_1 and O_2 are sets the general condition illustrated in Figure 2.9 is given by

$$O_1 - O_2 \neq \phi$$

---

* Here we mean an element in the good input sequence will be changed. Note that we are treating the output sequence as a member of a binary relation. We simply note whether any two sequences are the same or not the same.

where the term on the left is called the set difference. Note that $O_1 - O_2 \neq O_2 - O_1$ in general. ϕ is the empty set.

The idea of set difference can be useful in developing fault detection and isolation information. We proceed to describe the development.

The output sequences generated for a fixed input signal, fixed initial state (or initial conditions) and a fixed sampling interval but different parameter settings is written as

$$y_j(P_0, \underline{u}, t^*)$$

for the good parameter set and

$$y_j(P_r^q, \underline{u}, t^*)$$

for the rth parameter set to its qth value. Recall that $r = 1,2,3,\ldots,m$ and $q = \frac{-k}{2}, \left(\frac{-k}{2}+1\right), \ldots, -1, +1, \ldots + \frac{k}{2}$*. The range of q is k.

The expressions for the y_j during the t_ℓ th sampling interval and for the P_r^q parameter set is

$$y_j(P_r^q, t_\ell)$$

where $\underline{u}$ is understood to be invariant.

The sample of y_j during the t_ℓ th time interval is denoted by superscripting; $y_j^\ell(P_r^q)$. Finally, we can write, the family of all y_j sequences which are sampled during M

---

* To avoid confusion, it is assumed that the last k is always even.

intervals as

$$\left\{ y_j^1(P^0),\ y_j^2(P^0),\ y_j^3(P^0),\ \ldots\ldots,\ y_j^M(P^0) \right\} = Y_j^0$$

$$\left\{ y_j^1(P_1^{\frac{-k}{2}}),\ y_j^2(P_1^{\frac{-k}{2}}),\ \ldots\ldots,\ y_j^M(P_1^{\frac{-k}{2}}) \right\} = Y_j^{1,\frac{-k}{2}}$$

$$\vdots \qquad\qquad \vdots$$

$$\left\{ y_j^1(P_1^{-1}),\ y_j^2(P_1^{-1}),\ \ldots\ldots,\ y_j^M(P_1^{-1}) \right\} = Y_j^{1,-1}$$

$$\left\{ y_j^1(P_1^1),\ y_j^2(P_1^1),\ \ldots\ldots,\ y_j^M(P_1^1) \right\} = Y_j^{1,1}$$

$$\vdots \qquad\qquad \vdots$$

$$\left\{ y_j^1(P_m^{\frac{-k}{2}}),\ y_j^2(P_m^{\frac{-k}{2}}),\ \ldots\ldots,\ y_j^M(P_m^{\frac{-k}{2}}) \right\} = Y_j^{m,\frac{-k}{2}}$$

$$\vdots \qquad\qquad \vdots$$

$$\left\{ y_j^1(P_m^{\frac{k}{2}}),\ y_j^2(P_m^{\frac{k}{2}}),\ \ldots\ldots,\ y_j^M(P_m^{\frac{k}{2}}) \right\} = Y_j^{m,\frac{k}{2}}$$

Note that for each value of r, there will be k values of q. Hence the total number of sequences which can be generated, assuming all parameters are set one-at-a-time to the k possible values is mk. Each sequence has M elements.

To simplify the notation, the output sets can be

numbered sequentially from 1 to mk. This gives the family of output sequences

$$\left\{Y_j^0, Y_j^1, Y_j^2, \ldots, Y_j^n, \ldots, Y_j^T\right\} \qquad 2.36$$

where $0 \rightarrow 0$, $1 \rightarrow 1, -\frac{k}{2}$; $2 \rightarrow 1, \left(-\frac{k}{2} - 1\right)$; etc. where $\rightarrow$ means "corresponds to the term". $T \rightarrow m, \frac{k}{2}$.

The jth output co-domain for the network with a fixed input sequence and initial conditions is given by

$$\bigcup_{n=o}^{n=T} Y_j^n = \Delta_j \qquad 2.37$$

The range of any single sequence is simply the elements in the set Y_j^n. The co-domain for all outputs is Δ.

Two important theorems can now be stated.

Theorem 1

If the difference between all pairs of output sets Y_j^k for a particular output of necessary extent, formed from $Y_j^o - Y_j^k = D_j^{ok}$, $k = 1,2, \ldots, T$, is non-empty for all k, the parameter changes corresponding to each Y_j^k are detectable and the network faults are possibly isolateable for the set of conditions used to generate the Y_j^o and Y_j^k's. The conditions are: a fixed input sequence, fixed initial conditions, and fixed sampling interval. We abbreviate these as IICOS conditions from now on.

A proof requires the definition of a sequence of necessary extent. An output sequence of length M is said to be of necessary extent if for N faults, the equality

on the relationship

$$N \leqslant \sum_{n=1}^{M} \frac{M!}{n!(M-n)!} \qquad 2.38$$

holds. M is then the sequence length. We denote the solutions for M by

$$M = !N \quad . \qquad 2.39$$

Now, if $Y_j^o - Y_j^k$ for any k say $k = k'$ is empty, the sequence Y_j^o is equal to $Y_j^{k'}$ by the definition of set difference. Then the network fault corresponding to Y_j^k will not be detectable through output y_j because two output sequences for different parameter conditions are identical.

Hence, for the network fault to be possibly isolateable, the difference D_j^{ok} must be non-empty for <u>all</u> k. The value of M which is necessary and sufficient will depend on many factors. All that can be said in general is that for practical networks,

$$M \geqslant !N \quad .$$

From Theorem 1 we can state that if a given parameter change (fault) is undetectable, the output sequence obtained under the particular fault condition gives an output sequence equal to Y_j^o for a given output j.

<u>Theorem 2</u>

For a given network with fixed IICOS conditions, the fault corresponding to the kth parameter change is <u>definitely isolateable</u> if

$$D_j^{kh} = Y_j^k - Y_j^h \neq \emptyset$$

for all h = 0,1,2, ..., k-1, k+1, ..., T.

Again, a proof follows from the fact that if $Y_j^k - Y_j^h = \phi$ for at least one h, $h \neq k$, the parameter change giving Y_j^k is equivalent to that giving Y_j^h or in other words, the output sequences are the same for different parameter sets. Because of the method for constructing the Y_j^k, we conclude that if the difference is empty, the sequences are identical and the two parameter changes are equivalent for the given test. They may, however, be different from Y_j^o.

A useful corollary is the following: If $D_j^{kh} \neq \phi$ for all pairs of kh corresponding to the upper or lower triangular matrix and excluding the main diagonal, one-at-a-time network faults are definitely isolateable for output j. Hence fault isolation is possible for one-at-a-time faults. The maximum number of faults in the definitely diagnosable network that can be treated by this method is given by Equation 2.39.

Algorithms can be developed for fault isolation based directly on the technique of computing the output for each parameter value setting. This will be referred to as the "sequence element perturbation detection" (SEPD) method. If the assumptions which were made for the above development apply to the network being diagnosed, then this method offers a direct and useful approach.

One objection to the SEPD approach is that magnitude information contained in the sequence elements is only partially used. We specify only changes in level. If explicit magnitude information is used, it may be possible to shorten the test sequences. The methods developed in the next section are aimed at applying magnitude information to reduce the length of the output sequences and to refine the fault isolation schemes.

2.7 TEST ALGORITHMS BASED ON MULTIPLE LEVEL-TIME QUANTIZATION

In this section, the set difference method of the previous section is extended to utilize the magnitude information in the output signal. We assume that signals are sampled and that the values lie in a finite range of values which is determined by defining quantization levels. Properties of the spaces described by these signals are then exploited to develop algorithms for fault detection and isolation. We first introduce the planar signal space and then discuss the cubic signal space.

At this point, we reiterate that the aim of this thesis is to develop methods to diagnose hybrid networks. Obtaining signals in discrete form allows digital techniques to be universally applied to generation, reduction, and analysis of the signals. In fact, if the discrete approach is used, there is reason to believe that a category of a priori algorithms can be developed which will apply to either analogue or digital networks. [illegible] on this [illegible] is [illegible] 5.

2.7.0 Planar Signal Spaces

Diagnostic techniques based on a planar signal space definition are an extension of the discrete sampling method of the previous section. We begin the discussion by developing the planar space relationship between the parameter set $\{p_1, p_2, p_3, \ldots\ldots, p_m\} = P$, the input vector $\underline{u} = \{u_1(t), u_2(t), \ldots\ldots, u_l(t)\}$ and the output vector $\underline{y} = \{y_1(t), y_2(t), y_3(t) \ldots\ldots, y_n(t)\}$. A particular output will again be denoted by y_j. If the jth output, $y_j(t)$ is sampled at intervals of Δt, the output signal becomes a sequence y_j which will be finite.

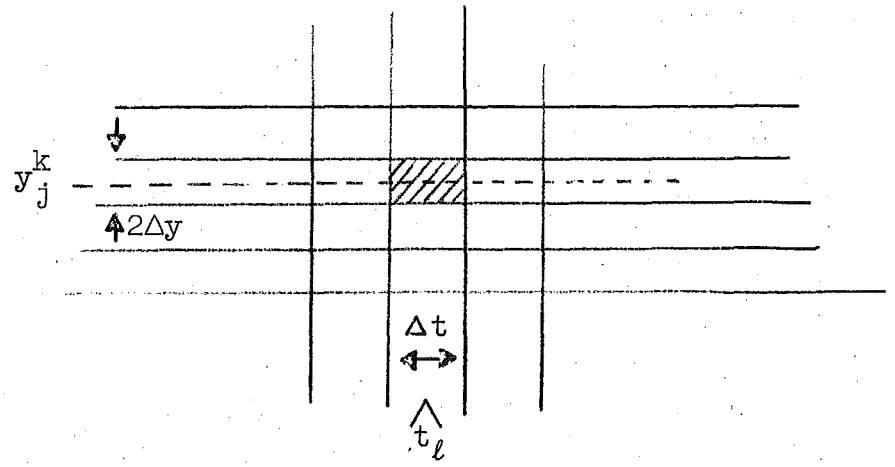
in length. Its cardinality always satisfies $\{y_j\} < \infty$.

Values of a sampled network output y_j can be obtained by a process which assigns to each value of t^* in the sampled time domain a value $y_j(t^*)$ in the co-domain. $t^* = t_i$ is a real number and satisfies $t^* = t_o + n\Delta t$, n integer. If y_j is sampled during instants Δt apart, the range of y_j will be a finite set if the number of samples is finite. If the range of the y_j is (y_j^{min}, y_j^{max}) and the instruments used to measure y_j are accurate to $\pm\Delta y$, then the number of levels, in the output range, ρ, is given by

$$\left| \frac{y_j^{max} - y_j^{min}}{2\Delta y} \right|$$

where $|\ \ |$ means "nearest integer".

If a sequence of output values has length M, the signal plane is said to have dimension ρM. A typical area (point) in the discrete signal plane is shown in Figure 2.10.



Section of Signal Plane
Figure 2.10

During any sampling interval t_ℓ, the uncertainty in

y_j is $2\Delta y$ and the uncertainty in time is Δt. The "area" of the discrete signal domain is given by

$$2 \cdot \Delta y \cdot \Delta t \cdot \rho \cdot M \qquad 2.40$$

For fault isolation, we are interested in determining how the change in particular parameter values affects the magnitude of the output in the t_ℓth time interval. To obtain this information, parameter value versus output value trajectory can be computed by incrementing the parameter(s) in very small steps and recording the resulting output values as a function of the parameter value. The solutions for each interval of time for all different parameter values are required. A typical output value versus parameter value for a single parameter and a sampling interval t_ℓ is shown in Figure 2.11.

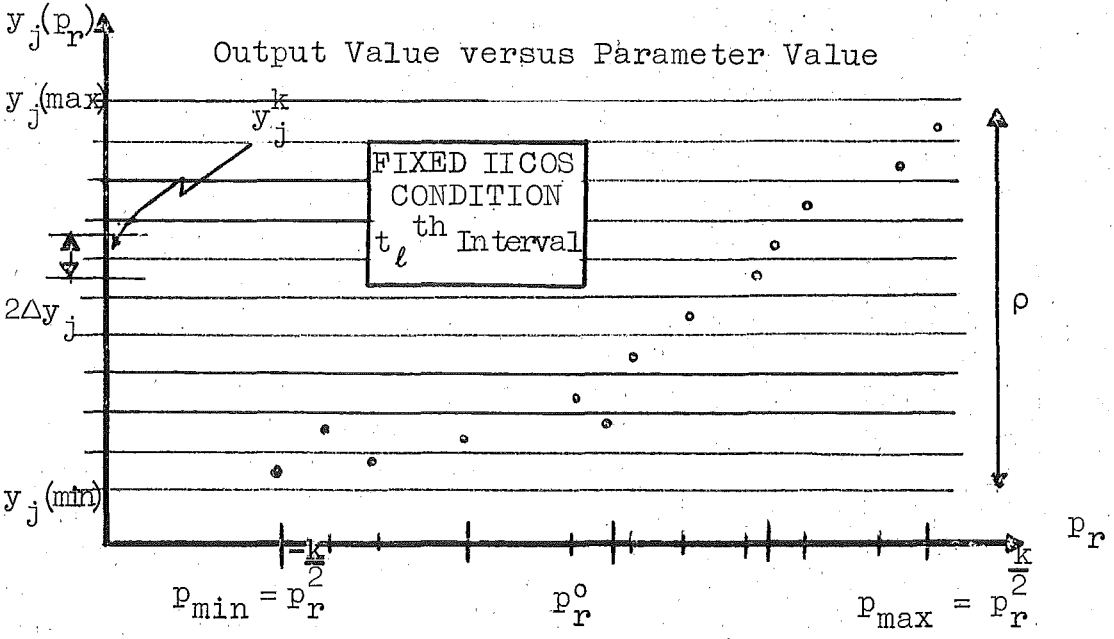


Figure 2.11

Thirteen parameter settings are shown and ten <u>different</u> y values which range from y_j^{min} to y_j^{max} result. The plot should show a continuous function but since discrete techniques are to be applied, values of the parameter

which change the output from one quantization level to the next are used. The minimum parameter change required can be used to set the lower limit on the step size to be used in the parameter variation simulation. However, another approach which varies the step size will be described later. The plot in Figure 2.11 for output versus parameter variation assumes that the parameter has an upper and lower drift limit. This assumption is based on the electronic component manufacturers practice of specifying a nominal value and the upper and lower drift limits. If catastrophic failures were to be treated by this method, parameters would range in value from 0 to ∞. There is no inherent limitation on the actual amount of parameter variation that can be treated by this method. However, limitations imposed mainly by computation time preclude the use of such a wide range of values. In addition, most networks "hang up" when one of the components fails catastrophically. The technique by Breen and Webb[142] mentioned earlier is designed to handle the catastrophic fault using linear d.c. or a.c. network models. Their method simulates this type of fault by setting parameters to very small values and to very large values and computes the output under these conditions.

Assuming one-at-a-time degradation failures only, information for a network output can be developed and presented in the planar space representation. This information can be applied directly to the test development problem. However, before proceeding with this objective, a variation in the presentation of signal information will be introduced. This variation will be termed the signal cube representation. It permits a geometric interpretation of the output signal information which is slightly more general and potentially more useful than the planar description.

2.7.1 Cubic Signal Spaces

It will have become apparent that diagnosis involves not only the generation, but also the storage and processing of large quantities of information. Methods which simplify any stage of the process are highly desirable. The cubic signal space representation possesses several attractive features. First, information is generated in the same form as required for planar signal interpretation. Second, geometric insight into the process of describing a fault can be gained. And third, the amount of information that must be stored can be substantially reduced.

Suppose that test for the network shown in Figure 2.12

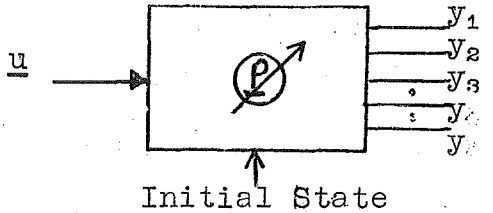
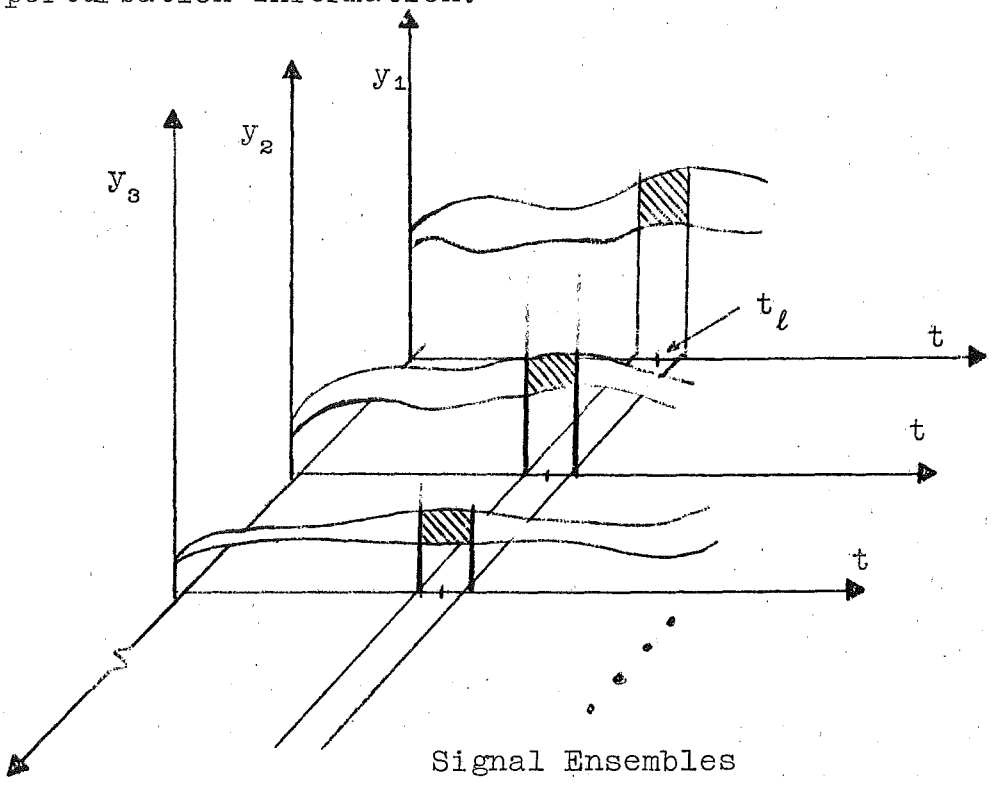


Figure 2.12

are to be developed. The parameters can be set to various values and the outputs recorded. Because the network is analogue, the outputs will be continuous functions of time. For one set of parameter values, y_n single trajectories will result. In general, there will be a different trajectory described by each y_j and for each different set of parameter values. For all sets of parameters in the network shown in Figure 2.12, an output trajectory envelope for each output will be described. The range of the output during any sampling interval t_ℓ can be thought of as occupying an area in the planar signal space. An example of this is shown in Figure 2.13. Three outputs are shown and the three ranges during the interval t_ℓ are cross hatched. For signal cube developments we will assume that the range is defined as

$[y_j(max)(t_\ell) - y_j(min)(t_\ell)]$. In Figure 2.13, the cross tched areas exaggerate the width of the sample taken at t_ℓ. We can now consider methods for condensing information in the quantized time-range form; this condensation will subsequently be related to parameter perturbation information.



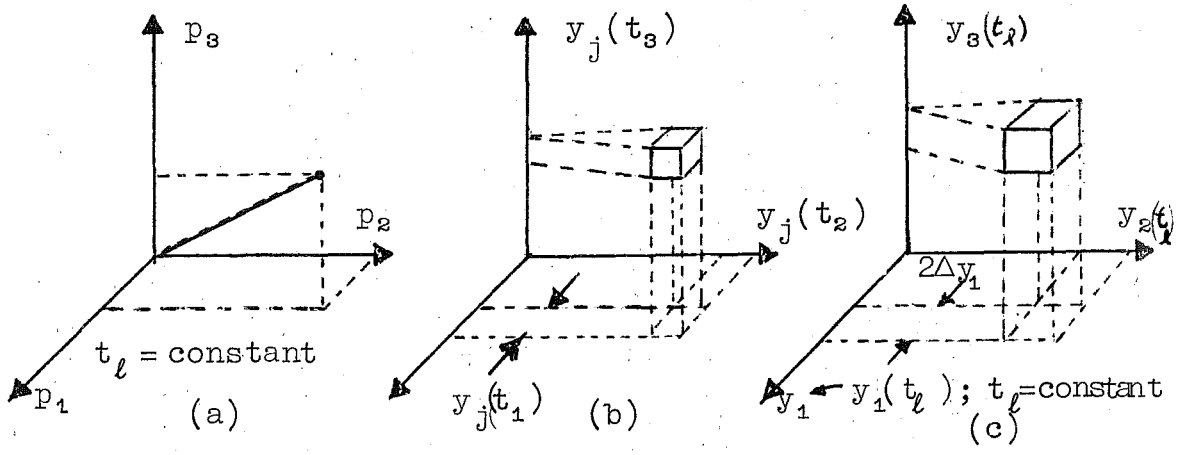
Signal Ensembles

Figure 2.13

2.7.1.0 Conversion of Sequences to Points

Any sequence containing M elements can be regarded as representing a point in M dimensional Euclidean space. The sequence elements may be ordered arbitrarily or by their natural appearance in time or space. By associating a co-ordinate with each element in the sequence, a cartesian co-ordinate system may be developed for the sequence. The parameter set can be condensed by similar methods. For example, Figure 2.14 shows three three-dimensional co-ordinate systems. Figure 2.14(a)

shows the parameter space which is defined by associating a co-ordinate with values of each of three parameters. It is assumed that the parameters are measured at one value of time t_ℓ and that the measurement is exact. A space having different properties is defined for the quantized-sampled output signal. In Figure 2.14(b), the cubic output signal representation for three different times is shown. It is assumed that the value is known only to within a quantization level. The quantization interval has a width of $2\Delta y$. If the sequence of points shown in Figure 2.8 were plotted on this co-ordinate system, the result would be a point. Because we are going to specify values only to within an interval, the signal is represented as a cube. Figure 2.14(b) could be described as representing the single <u>record</u> of one output signal of the type illustrated in Figure 2.13. If the <u>ensemble</u> value is desired for a given point in time, Figure 2.14(c) would be the appropriate description.



Parameter and Signal Description

Figure 2.14

The dimensions of these spaces can be increased to any positive integer number. If the signal value at each point in time t_ℓ, is confined to a point, the $\ell_{max} = M$

dimensional space will contain a point describing the set of samples such as shown in Figure 2.8. For the quantized signal spaces defined by the illustrations of Figure 2.14 (b) and (c), the description is in terms of a cube or hypercube. The dimensions of a cube may be regarded as a measure of the undertainty in the value of the actual point which lies some place within the cube.

2.7.2 Signal Cubes and Parameter Variations

We pause at this point to discuss the effect of noise on the output. The discussion is superficial. Because quantization of the signals is assumed, noise will change the instantaneous value of the signal but under normal conditions we assume that the effect will be to move the signal within the given cube. In essence, it is assumed that noise causes time dependent variations in the signal about the mean value which is y_j and that noise moves the value around within the interval $\pm\Delta y$. In other words, the expected value of the signal lies at the zero noise level between the limits of the quantization level. For a perturbation algorithm described in Section 2.8.2 using planar signal descriptions, the computations are assumed to be noise free. To apply information developed in a noise free environment to noisy a posteriori testing, averaging techniques will be required. If averaging techniques are not employed, confidence in results will not be high.

If a given parameter is continuously varied and the noise free network output is measured for each infinitesimal variation, then the mapping performed by the network will change as a function of the parameter setting provided the condition given by Equation 2.11 holds.

This change in mapping will be reflected as a shift in the signal cube. The shift will, in general, depend on the parameter being changed and on value of the parameter, all other controllable variables being held constant. The shift in the signal cube corresponds to a change in signal quantization level during one sample interval.

An interesting phenomenon occurs in the geometry of the output signal space. As the signal in one co-ordinate changes level, the entire cube (its faces) changes except for the face corresponding to the altered co-ordinate. Hence, given two adjacent n dimensional cubes which share a common face in the n dimensional Euclidean space, the co-ordinate axis intersected by a hyperplane formed by an extension of the face common to the two adjacent cubes corresponds to the signal which has changed to form the new cube. The co-ordinate axis value which is intersected will be called the face value.

2.7.3 Test Development Using Cubic Signal Spaces

The notion of the signal cube can be used to develop a fault detection and isolation scheme. Subsequently, we shall describe corresponding planar methods.

First of all, assume again that a given network with fixed IICOS conditions is prescribed and that the input and initial state combination excite all modes of the network. In addition, assume that perfect sampling which measures the signal at a fixed instant within the interval Δt and to within $\pm\Delta y$ can be obtained.

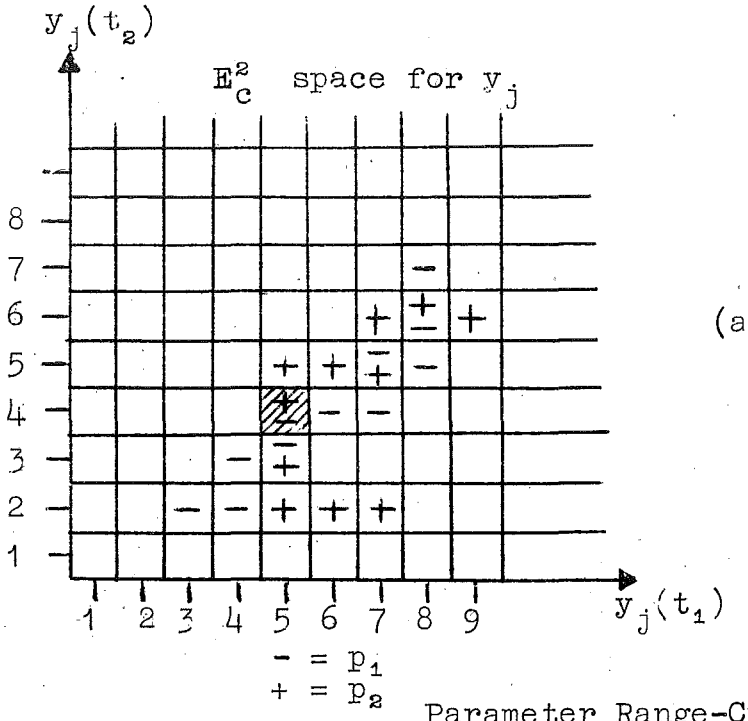
To develop diagnostic information using the cubic description, all parameters in the network model are first set to their nominal value and the time domain

solution at the prescribed sampling instants is obtained. The output signal values $y_j \pm \Delta y$ at each discrete time instant t* define the good cube in M dimensional space where M is the total number of samples. Next, the first parameter p_1 is varied continuously or in extremely small increments until the cube shifts in one co-ordinate. These shifts will preserve one face in common to the old and new cube. The value of the parameter p_1^1, and the face value at which the change occurred are recorded. The parameter is varied in the same direction until another change occurs and p_1^2 is recorded along with the face value. This procedure is continued until all parameters have been varied one-at-a-time through their respective ranges and the face value recorded for each. Using a signal cube description and noting that initial departures from the good cube (which is defined by the good output signal) may be different, we can state the following theorem.

Theorem 3

Given an analogue network containing m parameters and having fixed IICOS conditions, the minimum number of co-ordinates corresponding to M samples in the quantised cubic signal space E_c^m, required for fault isolation is $|m/2|$. A fault corresponds to any parameter value which causes the output to lie outside the design value in one (or more) co-ordinate position(s). $|\ |$ means "nearest integer".

The proof follows easily from the fact that if the signal space has dimensions M cubes have 2M faces. For fault isolation, the input-output mapping defined by the network must result in unique departure directions from the cube defined by the good output. It can only be unique for 2M faces. Hence the least number of co-ordinates is



(a)

Parameter Drift Table

$y_j(t_1)$	$y_j(t_2)$	Parameter Value
5	3	p_2^{-1}
5	2	p_2^{-2}
6	2	p_2^{-3}
7	2	p_2^{-4}
5	5	p_2^{1}
6	5	p_2^{2}
7	5	p_2^{3}
7	6	p_2^{4}
8	6	p_2^{5}
9	6	p_2^{6}
5	3	p_1^{-1}
4	3	p_1^{-2}
4	2	p_1^{-3}
3	2	p_1^{-4}
6	4	p_1^{1}
7	4	p_1^{2}
ETC.		

(b)

Parameter Range-Cube Co-ordinate Table

$y(t_1)$	$y(t_2)$	RANGE
5	3	$p_2^{-2} - p_2^{-1}$
5	2	$p_2^{-3} - p_2^{-2}$
6	2	$p_2^{-4} - p_2^{-3}$
7	2	$p_2^{min} - p_2^{-4}$
5	4	$p_2^{1} - p_2^{-1}$
5	5	$p_2^{2} - p_2^{1}$
	•	
	•	
9	6	$p_2^{max} - p_2^{5}$
ETC.		

(c)

Figure 2.15

m/2 or the nearest integer upward. In practical networks, deviations of a given parameter in one direction will usually cause a face value different from the face value due to changes in the opposite direction. Hence a more realistic condition is $M \geq m$. The lower limit (and the most efficient test) under this assumption would be $M = m$. This is of course not in general sufficient. This can be illustrated by the following example.

Figure 2.15(a) shows a network output y_j, sampled at two points in time t_1 and t_2. The network is known to contain two parameters. Deviations in parameter values cause changes in the output cube and the changes are recorded as outlined in the previous section. An example giving some of the possibilities is shown in Figure 2.15(a). The cross-hatched cube in Figure 2.15(a) indicates the good operating conditions and the deviations for p_1 and p_2 and the corresponding output values are coded by the symbols - and + respectively. The set of cubes intersected during the excursion of a given parameter from one of its extremes to the other defines a parameter path. The set of parameter co-ordinates versus value for any test is given in the parameter drift table shown in Figure 2.15(b). The superscript (-) notation of the parameters indicates values down from nominal. No sign indicates upward drift. Superscript "o" on a parameter denotes the nominal value. Using the parameter drift table, a parameter range-cube co-ordinate table can be compiled as shown in Figure 2.15(c).

The parameter range-cube co-ordinate table shows the parameter range over which the output cube remains invariant. This table is important because it enables the heretofore discrete techniques to apply to the quasi-continuous situation. The continuous variation is what occurs in reality. The discrete techniques are

used for computational convenience.

The information given in Figure 2.15(c) can be better appreciated by referring back to Figure 2.11. It was shown in Figure 2.11 that variations within certain intervals causes no observable effect in the quantized output signal. The reader may verify that in this figure, only 12 intervals are required to define the parameter values. This number is, of course, dependent on the quantization range $2\Delta y$. Now in Figure 2.15(c), information is developed by noting that the parameter value range over which the signal cube is fixed can be got from the parameter drift table. For example, the values of the parameter p_2 satisfying $p_2^{-2} < p_2 < p_2^{+1}$ gives an output signal whose coordinates are $[y(t_1), y(t_2)] = (5,3)$. This can be verified by checking the first row in the parameter range-cube co-ordinate table in Figure 2.15(c).

From this discussion and Figure 2.15, it can be seen for example, that if a parameter is changed to a value which causes the output cube to show "single occupancy" for example: (3,2), (4,2), (5,2), (6,2) or (7,2)*, then parameter values giving rise to outputs whose co-ordinates define this cube can be isolated from all others. For the five co-ordinates listed, the first two indicate p_1 faulty and the latter three implicate p_2. If a cube has "double occupancy", two possibilities exist and so on.

Intuitively, if no two parameter paths intersect in any one cube, the faults are isolatable for <u>all</u> parameter values. The fewer the number of intersections,

---

* ( ) denotes the Cartesian co-ordinates of a cube.

the greater the isolation capability. The parameter path and parameter drift table for a given network can be used to compute the testability for the network using the given test.

The testability measure for fault isolation considering one-at-a-time variations and given IICOS conditions and a cubic space procedure is defined by:

$$\text{Testability (I.F.C.)} = \frac{\text{Cubes intersected by a single parameter path}}{\text{Total cubes occupied}}.$$

I.F.C. means "Isolatable using Fault Cube procedures".

There are several ways to improve the detection and isolation using cubic reduction methods. The input may be changed (although this is not always possible for on-line methods), the input record may be lengthened, or the output dimensions may be increased. It may be necessary to use a combination of these to effect an improvement.

A different interpretation of the quantized signal space approach permits test methods to be extended to fault detection. The test is based on the binary relations: $p_i < p_i$ (low) or $p_i > p_i$ (high). *

For a particular test if a fault causes the signal cube to change in one co-ordinate position, the fault is detectable. Suppose that $p_i > p_i$ (high) and $p_i < p_i$ (low) cause output values outside the good trajectory range. By definition, a fault corresponds to the conditions $\{p_i(\text{low}) > p_i\}$ or $\{p_i > p_i(\text{high})\}$. Then any $p_i > p_i$ (high) or $p_i < p_i$ (low) which causes the output cube to be different from the range of good cubes is a detectable fault. The fact that a fault is detected will be denoted $[\text{F.D.}p_i(\text{low}) = \text{TRUE}]$ for all faults $p_i < p_i$ (low) and

* low ≡ min and high ≡ max in Figure 2.15.

[F.D.p_i(high) = TRUE] for all faults $p_i > p_i$ (high). Then we have the following definition:

$$\text{Testability (D.F.C.)} = \frac{\sum_{i=1}^{N} \text{F.D.}p_i(\text{low}) + \text{F.D.}p_i(\text{high})}{2N}$$

where N is the number of parameters. If F.D.p_i(•) is true, a 1 is added to the numerator, otherwise nothing is added. D.F.C. means "Detectability using Fault Cube procedures." Fixed IICOS conditions are assumed and the SEPD method is used to obtain the information.

2.7.4 An Algorithm for Computing Parameter Paths Using Perturbation

It is assumed that a network model is available and that the IICOS conditions can be stipulated. Using the following algorithm, information for the signal plane and signal cube test methods can be developed using a digital computer program. Parameter values are varied in discrete amounts and output signal information is quantized.

Algorithm

1. Set all network model parameters to nominal (mean) values.
2. Solve for output values at discrete times t^*. Space sampling intervals Δt apart. Find M samples for each output.
3. Quantize output values by one of the following methods:

   (a) Use a pre-selected quantization $2\Delta y$. If y (interval time = (k)) is $[y^q - \Delta y] \leq y \leq [y^q + \Delta y]$, then set $y = y^q$. q denotes quantization level.

OR

(b)(i) Compute $[y(t) - y(t-1)]$ and $[y(t) - y(t+1)]$. Are both negative? If 'yes', call large one $-\Delta y(t)$. If one is negative, call it $-\Delta y(t)$. If both are positive, use pre-selected Δy_i.

(ii) Repeat computation to determine upper quantization level. Use $[y(t-1) - y(t)]$ and $[y(t-1) - y(t)]$. Repeat three 'If' computations in 3(b)(i) except look for positive terms this time.

4. This completes the quantized planar space description. Note that for computation 3(b), the space is not described by equi-dimensioned rectangles. Ordinarily, the quantization will be dictated by measurement accuracies. Hence, 3(a) will be preferred.
5. Set the first parameter to a positive deviation $p_1^o + \Delta p$.
6. Compute the output(s) using the same sampling intervals selected in 2.
7. Record the quantization level-time interval pairs which are <u>different</u> from the mean values.
8. Successively step p_1 to its maximum limit, computing the output for each value and recording the outputs which are different from the mean and from any previous computation. (Note that unless 3(a) is used, an elaborate method will be needed to keep track of the non-uniform quantization.)
9. Repeat 5-8 for each parameter, recording [quantization level-interval-parameter value] triples.
10. Perform the 5-8 computations but in this case, use $p_i - \Delta p_i$ values.

The information generated can be used for a parameter path study. The information is stored in an array as follows:

PARAMETER SETTING	$\bar{t}_\ell$	OUTPUT VALUE - INTERVAL, ℓ						
		1	2	3	4	5	...	M

Note that for a network with 50 parameters, using 20 variations and 50 time intervals, 50,000 storage locations would be required for the $50 \times 20 \times 50$ array to hold the information. However, only a few outputs will deviate from the previous value so many of the rows in the above table may be empty. A dynamic storage scheme can be used to limit the memory requirements. There are various ways to reduce the storage requirements even more. For example, the array may be a linear array of words, sufficient in length to hold 8 place integer digits. By coding the outputs on a relative-to-nominal basis, (assuming fixed quantization levels), a typical code could be

2	5	5	0	4	4	7	0	3

Parameter Setting O/P Value O/P Number Time Interval

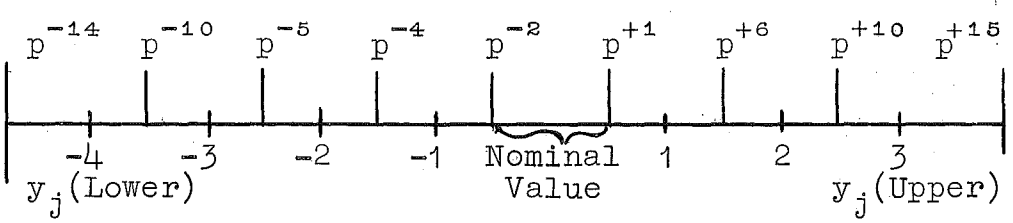
which would mean: the 25th parameter set to its nominal setting (50 indicating nominal), output value 6

quantization levels down from nominal (44), output number 7 during the 3rd time interval. Using this "relative to nominal" scheme, large amounts of information can be stored efficiently in integer format.

Using the quantized signal space approach in conjunction with information generated by the above algorithm, parameter paths and parameter path intersections can be found. In addition, a fault dictionary can be developed using different arrangements of the above information.

A change in the variation routine which is useful for generating fault isolation information can be described as follows.

Each time a parameter is varied, the output sequence is checked to see if more than one level has changed. If two or more levels have changed, the value is decreased incrementally until just one level changes. If no changes occur during the first variation, obviously the parameter value will be increased. Considering again only one-at-a-time variations, it is possible to develop the hypercube departure planes mentioned in Section 2.7.3. This permits the application of discrete signal technique to continuous changes in the parameter values. For example, a single output deviation versus parameter value might be the following



This plot shows that if y_j is in level 2, the parameter value lies in the range p^{+6} to p^{+10}. In this scheme,

the parameter value changes may occur on the continuum except at the borders of the quantization levels.

2.8 TEST DEVELOPMENT USING PLANAR SIGNAL SPACES

The cubic method is preferred for developing diagnostic information in most cases. It leads to economic methods for storing the test information. However, using planar descriptions of Section 2.7.0, fault detection and isolation schemes may be developed. Methods implied by Theorems 1 and 2 do not use amplitude information. Only the change in a particular signal from the good value during a specified sampling interval is considered*. This method requires very little information storage when used for fault detection. A time domain signature technique will now be described similar to the one due to Berkowitz in the frequency domain. This will be followed by a second method called the modified set difference which uses the amplitude information and applies the set difference definition described in Section 2.7.1. A testing diagram is used to reduce the information to obtain fault isolation information. The reduction can be considered as a sequential or serial test.

2.8.0 Tabular-Signature Method

The method for reducing and displaying the multiple output network trajectory information in tabular form is best illustrated with an example. A table is developed like that shown in Figure 2.16. Parameters are set one-at-a-time to their maximum and minimum values.

---

*It was mentioned previously and should be reiterated here that the tests result in a decision which corresponds to a binary relation. (See Chapter 1, Section 1.7.2.2)

Multiple Output Trajectory Display

Parameter Value	Time Interval	y_1	y_2	y_3	. . . . .	y_m
p (MAX)	t_1					
	t_2					
	t_3					
	⋮					
	t_M					
p (MIN)	t_1					
	t_2					
	t_3					
	⋮					
	t_M					

Figure 2.16

The outputs are described as trajectories in the discrete signal space for sampling times t_1 to t_M, and are similar to those shown in Figure 2.13. The signature used in the table is developed by listing the values for y_j for a given parameter setting at times t_1, through t_M. Once the table is developed, it can be ordered according to the decimal magnitude of the number formed by the signature $\{y_j(t_\ell)\}$. The signature and associated parameter setting for each interval and each output can be stored on magnetic tape or disc and later interrogated during a posteriori diagnosis. The procedure describes a combinational test followed by a comparison.

The application of this method requires the assumption that in the real system, the parameters will drift one-at-a-time. By comparing signatures which are compiled during the a priori stage with those obtained for an actual network, the most likely fault can be selected. If this method is applied to fault isolation, when used in conjunction with parameter value

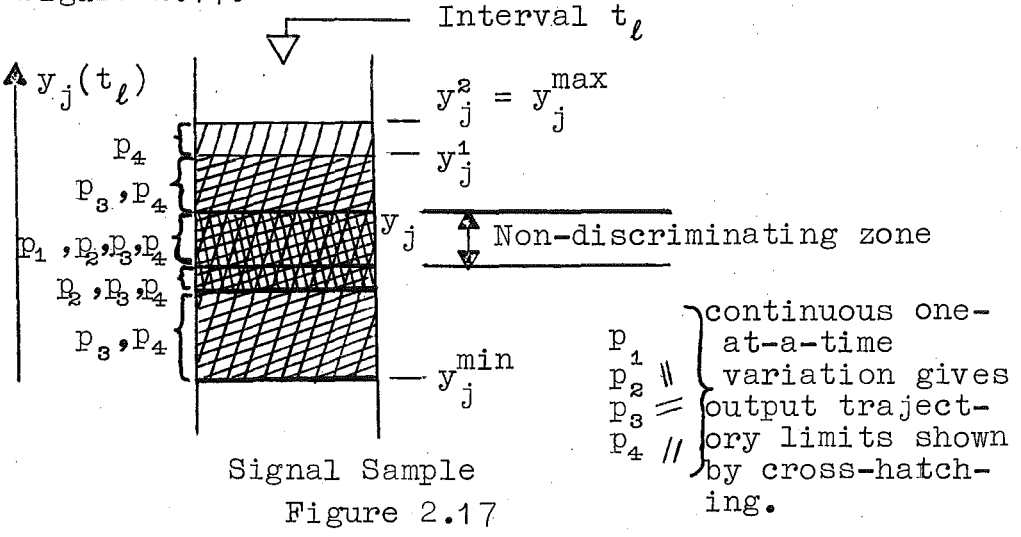
distribution information[63,81], the confidence in a decision that a particular parameter is responsible for a given output can be made quantitative. In addition, finer increments in the parameters, say $p_o - \Delta p_o \leq p \leq p_o + p_o$ are used, the fault resolution of the method will be improved. However, the method described in the next section which uses a multiple step decision process is believed to proffer the best fault isolation method if continuous changes in parameters are assumed to prevail.

2.8.1 Modified Set Difference Method

Fixed percentage changes in the parameters will cause a shift in the trajectory by different amounts for each parameter setting during any time interval. One measure of the trajectory change might simply be $[\Delta t \cdot (y_j^{max} - y_j^{min})t_\ell]$. Another and perhaps more useful measure of the change in trajectory with changes in parameter value is $y_j^o \cdot (y_j^{max} - y_j^{min})$ which can be regarded as the moment of the area about the time axis. y_j^o is the expected value of y_j for parameter set P^o. The measure of parameter influence that will be applied here is simply the difference in the min-max value obtained for all given parameter pairs. This approach will now be clarified by using a geometric interpretation.

Any two parameters which are varied through their respective ranges and which give identical trajectory ranges $(y_j^{max} - y_j^{min})$ for all j and in all time intervals t_ℓ cannot be distinguished by information from the outputs. However, if the output values are different for a small range of the two parameter perturbations, it may be possible to develop a method for discriminating between the two. Suppose that a parameter domain contains a range of values which gives a sub-set of

trajectory values $\{y_j(P_r^q)\}$ which are different from all others for a particular output and during a particular time interval. The fault corresponding to the range of values is definitely isolateable. A possibly isolateable fault (as used in this context) can be isolated if a testing diagram is developed over several outputs and for several time intervals. The idea can be illustrated by first considering the signal description shown in Figure 2.17.

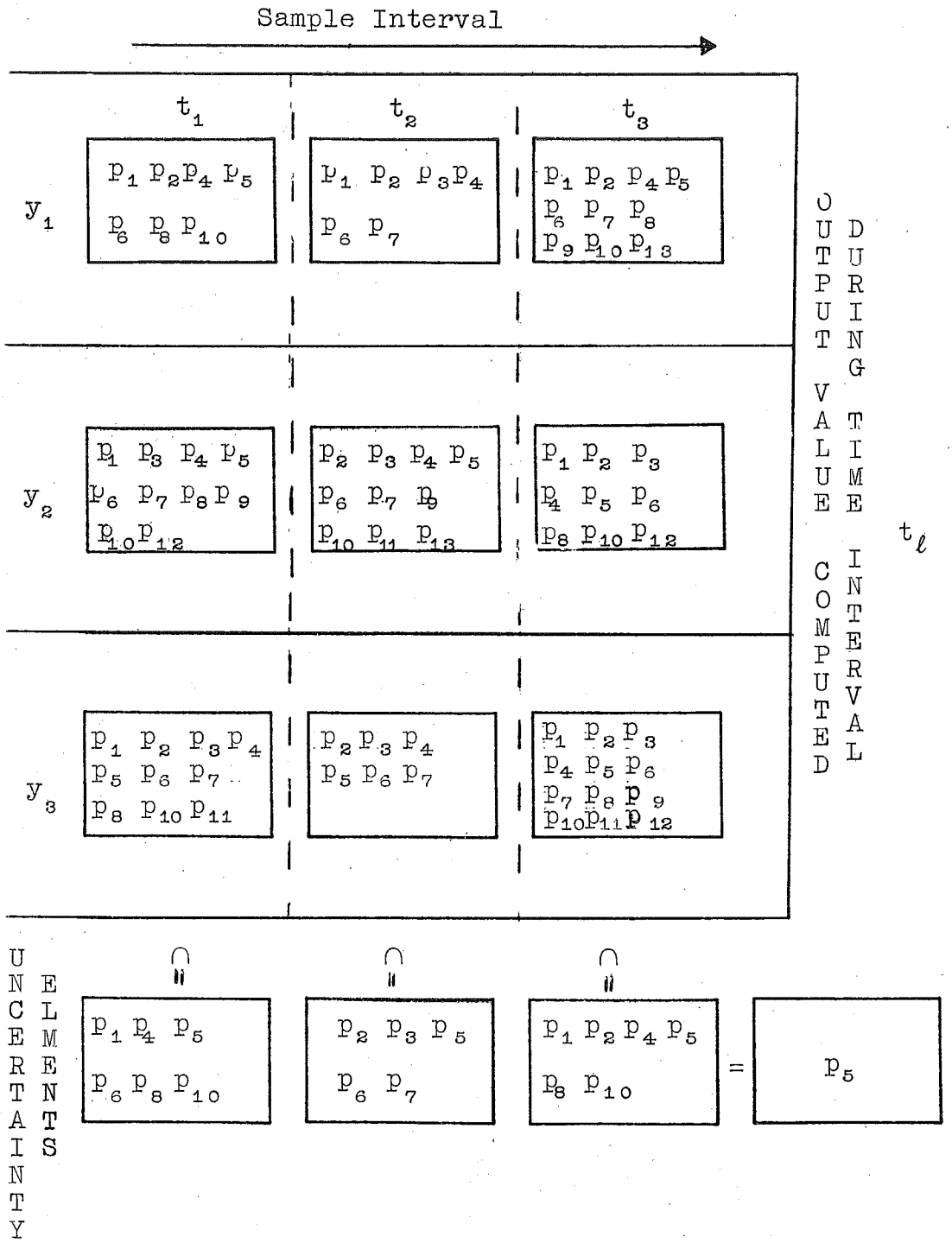


Signal Sample
Figure 2.17

This figure shows the min-max trajectory limits for an output y_j during the time interval, t_ℓ. For clarity, the effect of only four parameters is shown and the ranges corresponding to respective parameter ranges are cross-hatched or shaded. The good trajectory lies within some discrete level or range of levels in the non-discriminating zone. The non-discriminating zone covers all parameters and variations within this zone and cannot be attributed to any single parameter. However, it can be seen that p_4 values may be held responsible for values of y_j in the region between y_j^1 and y_j^2. Similarly, values of the output lying in the range y_j to y_j^1, may be attributed to either p_3 or p_4. There will be a similar picture for each time interval t_ℓ.

Assuming that several outputs are available and that the output sequence is of sufficient length, a testing diagram can be used to develop a decision algorithm to isolate the parameter(s) responsible for a fault (or faults) provided a) the value of some of the outputs lie outside the non-discriminating zones and b) the testing diagram terminates in a non-empty set. If the terminating element of the testing diagram contains a single element, the fault may be isolated to a single parameter. If it contains more than one element, then any of the elements in the set may be held responsible. If the probability of the parameter drifts are known, this information can be used to develop a listing of the "most likely cause" elements.

The idea of the testing diagram can be illustrated by the example shown in Figure 2.18. Each box shows the parameters which will cause the particular output to be the given value during the time interval t_ℓ. The parameters shown in each box can be regarded as representing the uncertainty that a certain output y_j during t_ℓ is due to a parameter setting in the range of allowable values for p_r. The testing diagram represents the serial decision process for reducing the uncertainty of which parameter (or parameters) is responsible for giving the values for each of the three outputs during the three time intervals shown. The intersection of the likely parameters is taken during each interval to narrow down the suspects. In addition an intersection is performed over the range of samples and the terminating element(s) is the parameter or parameters which has caused the variation. In Figure 2.18, the test terminates in the decision that p_5 is responsible for the nine output signal samples obtained. Kautz[69] and Brule et.al[19] have good discussions of testing diagram applications.



$$\text{Decision} = \bigcap_{\ell=1}^{3} [p_i(y_1, t_\ell) \cap p_i(y_2, t_\ell) \cap p_i(y_3, t_\ell)]$$

Figure 2.18

2.9 PSEUDO CORRELATION

2.9.0 Introduction

Chronologically in this study, pseudo correlation was the first network diagnostic technique to be investigated. It is essentially a method for encoding output signal information obtained from a network model. The model must be useful for simulating both fault-free and faulty modes of operation. The encoding is achieved by a transformation on the sampled output signal record and can be regarded as a mapping from a sequence of real numbers to a single real number. This many-to-one type of transformation bears some resemblance to cross correlation. For this reason, it has been termed pseudo-correlation. A set of pseudo-correlation values generated by this method a priori is useful for fault detection and for indexing a fault dictionary[137] for fault isolation. The a posteriori values are measured and compared with a priori generated values until a match is found; decoding of the information using a dictionary scheme is thus accomplished.

2.9.1 Basic Discrete Signal Technique

An analogue network having fixed IICOS conditions will have an output signal $\underline{y}(\underline{u},\underline{P},\underline{t})$ which is $n \times 1$. Considering a single component of the vector say $(y(\underline{u},P,t)$, or $y_j(\underline{u},P,t)$ or $y)$, we will multiply this component by w^n to obtain $w^n \cdot y(\underline{u},P,t)$, where $w > 0$ and integer and $n = 0,1,2,3,\ldots,M$. If the output signal is sampled every Δt seconds and the product of $w^n \cdot y$ summed for M samples, a value $\psi(M)$ is obtained. The operation will be written

$$\psi(M) = \sum_{n=0}^{M-1} w^n y(\underline{u},P,n\Delta t) \qquad 2.41$$

and the function $\psi(n)$ is termed the pseudo correlation function. Without any loss of generality we will write,

$$\psi(n)\Big|_P = \sum_{n=0}^{M} w^n y(n) \qquad 2.42$$

where $M+1$ is the actual number of samples including the sample at $n=0 \Rightarrow t=0$. The value of the pseudo correlation is written $\psi(M)$ - argument substituted - and the function is written $\psi(n)$. We will sometimes use $\psi(M,w)$ to show explicitly, dependence of the value of ψ on M and w.

The value of the pseudo correlation depends on $\underline{u}$, P, n and Δt. If $\underline{u}$ and Δt, M and the initial state are always the same for each computation, for fixed M, $\psi(M)$ varies with P only. The value of the pseudo correlation with all parameters set to good conditions is written $\psi(M)\Big|_{P^o}$. A necessary condition for fault detection is $\psi(M)\Big|_{P^o} \neq \psi(M)\Big|_{P\neq P^o}$. We note that there may be many (finite) M's or no M which will satisfy this condition. An important practical consideration is that the magnitude of the difference between $\psi(M)\Big|_{P^o}$ and any other value say, $\psi(M)\Big|_{P_r^q}$ i.e. $\psi(M)\Big|_{P^o} - \psi(M)\Big|_{P_r^q}$, must be sufficiently large to be resolved by computing methods developed for the digital computer. Equally important, the difference must be large enough to be resolved on measuring equipment that will be used in a posteriori diagnosis.

An example will illustrate the procedure for developing pseudo correlation which can be used for fault detection and isolation. For this example, we will suppose that a) the sampled analogue output signal from a network is quantized into two levels 0 and 1 and that b) the occurrence of a fault causes at least one quantization

level to change in one or more sampling intervals. The availability of a model of the network which can be used for parameter variation to simulate faults is implied. The assumption that a fault will cause the output value during at least one sampling interval to change if only two levels are used is probably fallacious for most real analogue networks, particularly if faults are assumed to be due to component parameter drift. This assumption becomes less objectionable for an increasing number of quantization levels. For present purposes, the reader may regard the two level assumption as a convenience which has been adopted to simplify the illustration.

Figure 2.19 shows a plot of the good output $y(\underline{u},P^o,t)$ obtained from the network. The signal is sampled at 13

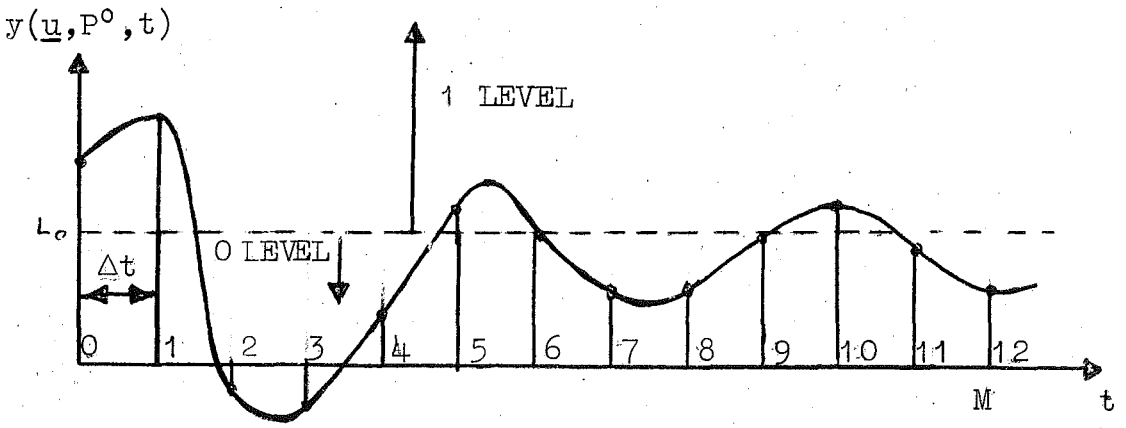


Figure 2.19

points in time at intervals Δt. Using delta function notation[14 8] we can write an expression for the good sampled output signal as

$$y(\underline{u},P^{o},t^{*}) = \sum_{n=0}^{12} \delta(t - n\Delta t)y(\underline{u},P^{o},t) \qquad 2.43$$

where $\delta(t - n\Delta t)$ is the delta function taken at time $n\Delta t$. The quantized sampled output signal is given by

$$\hat{y}(\underline{u},P,n) = \begin{cases} 1 & \text{for } y(\underline{u},P,t^{*}) \geq L_{o} \\ 0 & \text{for } y(\underline{u},P,t^{*}) < L_{o} \end{cases} \qquad 2.44$$

where L_{o} is called the quantization threshold and may be taken as the average value of y, or selected by some other suitable criterion. For example, it may be selected to give the most zero crossings over the sampling period, M. For the signal in Figure 2.19 we can write

$$\hat{y}(\underline{u},P^{o},n) = 1100010000100; \quad n = 0,1,2,\ldots,12 \qquad 2.45$$

Computing the pseudo correlation for $\hat{y}(\underline{u},P^{o},n)$ using $w = 2$ and interval Δt we obtain using Equation 2.42

$$\psi(n) = \sum_{n=0}^{M} 2^{n}\hat{y}(P^{o},n\Delta t), \ (\underline{u} \text{ dropped})$$

which gives

$$\begin{aligned} \psi(12) &= 1\cdot 2^{0} + 1\cdot 2^{1} + 1\cdot 2^{5} + 1\cdot 2^{10} \\ &= 1059 \end{aligned} \qquad 2.46$$

A convenient method for recording this result is to write: $(P^{o},1059)$, which can be used to mean: For the fault free system, the pseudo correlation is 1059.

For each possible parameter condition corresponding to a fault, P_{r}^{q}, the pseudo correlation can be computed and a list of fault-pseudo correlation pairs results. If the assumption that a fault causes a change to occur

in at least one quantization level is valid, then for N faults there will be N ordered pairs $(P_r^q, \psi(M)\big|_{P_r^q})$. The value of the correlation will be different from the value of $\psi(M)\big|_{P^o}$ by the original assumption, b). Fault detection is automatic. $\psi(M)$ is computed a posteriori from data obtained for the network whose state of repair is uncertain. If the value-computed (a priori) or (a posteriori) - is different from 1059, the network is judged to be faulty.

Fault isolation is accomplished by using the pseudo correlation values on a direct comparison basis. For a system in which a fault has been detected, a comparison of the measured-computed pseudo correlation with the a priori computed values is made until the two equal values are found. If more than one of the computed values is the same, then for the particular IICOS conditions, single fault resolution is not possible. The testability measures for Fault Detection and Fault Isolation for the pseudo correlation method can be written

$$\text{Testability (F.D.)} = \frac{\psi(M) \text{ Values Different From } \psi(M)\big|_{P^o}}{N}$$

$$\text{Testability (F.I.)} = \frac{\text{Total Different } \psi(M) \text{ Values}}{N+1} \qquad 2.47$$

where N is the number of faults. The value of the testability will depend on the IICOS conditions, on the number of quantization levels, and on the number of samples, M, used to compute ψ.

2.9.2 Remarks on Example

The method used in the illustration in Section 2.9.1 can be generalized to handle output information quantized

into a number of levels greater than 2. In general, the number of unique values for $\psi(M)$, (Equation 2.42) is L^M where L = w is the number of quantization levels and M is the number of samples. In the previous example, there were 2^{13} different possible pseudo correlation values. This corresponds to being able to distinguish a potential maximum number of N = 4095 different faults. In practice, the number which can actually be isolated will be less than theoretically predicted because one fault may give a number of different values (e.g. a drift fault) or resolution will be impaired by a condition where different faults give the same pseudo correlation value (e.g. catastrophic faults).

The value of the pseudo correlation $\psi(M)$ using w^n is integer provided w is integer and y is integer. If w = L, L integer, and L gets reasonably large, say 5, for example, then for a particular pseudo correlation encoding which uses M = 20, there are potentially

$$5^{20} \approx 95,370,000,000,000$$

different possible unique integer values. If these values are written on magnetic tape at about 1000 numbers per inch, it will take roughly 150,000 miles of tape. Sorting these presents some difficulties!

There are practical, theoretical, and computational problems which limit the efficacy and impede the development of pseudo correlation methods for diagnosis. The practical limitation in developing pseudo correlation information is related to the inherent accuracy of the model. The a posteriori practical limitation is one of measuring the actual system to the required accuracy. The a priori limitations may be regarded as model specification problems. To develop a particular

pseudo correlation it is necessary to select the IICOS conditions, the number of quantization levels used, and the number of samples M to be used. Basically, for a given network with given IICOS conditions, we would like to know: What are the optimal values for L and M? The answer to this will, of course, depend in a complex way on the particular network's characteristics. The second problem is that of computing large numbers. If the computer used to compute $\psi(M)\big|_P$ has registers of length λ, integer number computations in the computer are confined to $\pm 2^{\lambda} - 1$ where λ is typically 12,16,24,32,36, 48 or 64 for different computers. To isolate N faults requires N + 1 unique pseudo correlation values. To obtain this number for a register length λ requires the condition

$$2^{\lambda} \geq L^{M} \geq N \qquad 2.48$$

where L is the number of quantization levels and M is the number of samples.

2.9.3 Generalized Pseudo Correlation

The definition for pseudo correlation can be generalized by letting w be something other than an integer exponential function. In general, we can write (assuming a sampled output)

$$\psi(t^*)\big|_P = \sum_{M+1 \text{ samples}} w(t^*)y(\underline{u},P,t^*).$$

or for fixed interval samples

$$\psi(t^*)\big|_P = \sum_{n=0}^{M} w(n\Delta t)\delta(t - n\Delta t)y(\underline{u},P,t)$$

from Equation 2.43. Altering the notation slightly we get

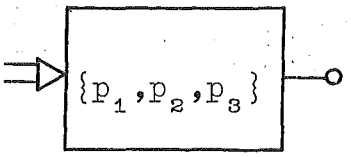
$$\psi(n)\Big|_P = \sum_{n=o}^{M} w(n\Delta t)y(\underline{u},P,n\Delta t) \qquad 2.50$$

where $w(n\Delta t)$ can be regarded as a discrete but not necessarily integer function of the $n = 0,1,2,\ldots$. The output is sampled at the same time and rate as we. We do not necessarily assume that y is quantized. An example which for illustrative purposes quantizes the output signal to the nearest integer value can be used to illustrate the method for developing a w function which is useful for computing $\psi(M)$.

The simple three parameter network is shown in Figure 2.20(a). It is assumed that each parameter has 3 fault settings and one good setting. Parameter settings (faults) are as usual assumed to occur one-at-a-time. The output, quantized to the nearest integer, for both the good and bad networks is shown in Figure 2.20(b). The value for the sum Σ can be regarded as $\psi(2)$ with $w(n\Delta t) = 1$ for $n = 0,1,2$.

Notice that the Σ values in Figure 2.20 have 3 ticks (rows 5,8 and 9) in the column indicating that only three of the values are the same as previous row values. In this case, computing Σ enables a fault to be isolated in all but four cases. In these cases, the fault is one of two possible faults. The remaining fault row 8, gives an output the same as the good output.

A weighting function w can be constructed and incorporated to improve the fault isolation capability for the particular IICOS test conditions. If for example, we use a w defined by $w(n\Delta t) = \{4,1,1\}$; $n = 0,1,2,$



NETWORK
(a)

	PARAMETER SETTING	Σ	OUTPUT SEQUENCES VALUES $y_1(\underline{u}, P, n\Delta T)$		
G	P^0	15	2	5	8
1	P_1^1	14	3	6	5
2	P_1^2	10	3	6	1
3	P_1^3	13	3	2	8
4	P_2^1	9	4	4	2
5	P_2^2	10 ✓	2	5	3
6	P_2^3	17	4	5	8
7	P_3^1	12	4	6	2
8	P_3^2	15 ✓	2	5	8
9	P_3^3	14 ✓	1	5	8
SAMPLE			n=0	n=1	n=2

Pseudo correlation for network with $w = 1(n\Delta t)$ and $\Sigma = \psi(2)$

Figure 2.20

and weight each sequence element by the appropriate w, we obtain the $\psi(2)$ versus P table shown in Figure 2.21. It is developed from Equation 2.50 by applying the information from the table in Figure 2.20 along with the specified w function. Using $w(n\Delta t) = \{4,1,1\}$,

	G	1	2	3	4	5	6	7	8	9
P	P^0	P^1_1	P^2_1	P^3_1	P^1_2	P^2_2	P^3_2	P^1_3	P^2_3	P^3_3
$\psi(2)$	21	23	19	22	18	16	29	24	21	17

Figure 2.21

the values for $\psi(2)$ are all different except for the fault corresponding to P^2_3. Because the y values for P^0 P^2_3 are identical, it will never be possible to select a w to distinguish these two cases using the three output samples only.

Heuristically, the larger the number of output samples, the greater the amount of information about the network or process. No good rules for selecting the number of output samples required to obtain a discriminating set of $\psi(.)$ values have been discovered. This is because the number of levels considered is not limited as in the planar and cubic methods but depends on the network. If the network is inherently level limited, for example: a clipping network, a lower limit on the number of required levels can be obtained using Equation 2.48.

The reader will appreciate that pseudo correlation using the general $w(n\Delta t)$ does not depend on pre-specified quantization levels. Rather it uses the unconditioned values of the outputs for all possible fault conditions to compute a quantity whose value depends on the order or the output values and on their magnitude. The quantity which we denote by $\psi(M,w)\big|_P$, depends on

specifying a $w(n\Delta t)$ which operates of the sequence element values to perform a type of expansion mapping. If $\psi(M,w)\big|_{P_r^q}$ is different from $\psi(M,w)\big|_{P^o}$ for all P_r^q, then $\psi(M,w)$ is useful for fault detection studies. If $\psi(M,w)$ has $N+1$ different values where N is the number of faults, then it is useful for fault isolation.

The generalized pseudo correlation method is an alternative to the cubic and planar methods. Using w^n weighting functions or the cubic and planar methods can lead to the computation and storage of very large numbers. The computations may not be compatible with computing equipment. However, generalized pseudo correlation does not contain this inherent limitation. It does however, require a method for selecting the weighting function $w(n\Delta t)$.

2.9.4 w Selection Algorithm

An algorithm for finding a w function useful for constructing a $\psi(n)$ function which will discriminate between various parameter settings in a network model will be described. Implicit is the assumption that the output terminal(s), structure, initial state and input of both the physical network and its model are fixed. The parameters are varied to simulate fault conditions by using fixed variations or by applying the perturbation algorithm described in Section 2.7.1.

The basic idea of the algorithm is to fix the initial weighting function to some known sequence of values, usually all ones. For a fixed output sequence length, and a given w, the value of ψ for each parameter set is then computed. For fault detection, the difference between $\psi(M,w)\big|_{P^o}$ and all other values is computed. If in each case the difference is sufficiently large (large

enough to be measured by ordinary techniques), the pseudo correlation function for the set of faults corresponding to the set $\{P_r^q\}' \subset P_r^q$ can be used for a posteriori purposes. If fault isolation is the aim, all $\psi(M,w)$ values must be unique. In either of these two cases, if the required conditions are not met, a new w must be tried and the resulting values again tested for uniqueness. We will describe the procedure for developing a ψ function for fault isolation; fault detection is a special case.

Suppose that a $\psi(n,w)$ function is developed which gives two or more values which are the same. For the two or more ψ values which are the same, recompute the outputs for these faults only. Call n' the first interval at which all the outputs differ. Set $w(n'\Delta t) = w'$ to a new value which amplifies the difference between the set of $y_j(n')$ values. As an illustration, suppose that three of the output values which we will denote by $y_j(n')_1$, $y_j(n')_2$ and $y_j(n')_3$ correspond to three fault conditions that give identical $\psi(M)$. New values for the $\psi(M)$, call them $\psi'(M)$ which are different from the first three $\psi(M)$ can be constructed using a technique illustrated in the diagram in Figure 2.22.

Assume without loss of generality that

$y_j(n')_1 = 1$

$y_j(n')_2 = 2$

$y_j(n')_3 = 3$

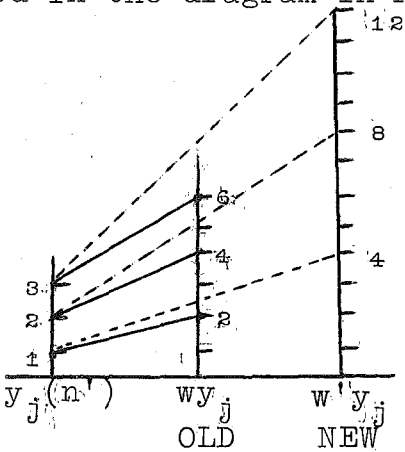


Figure 2.22

In this diagram, values for $y_j(n')_1$, $y_j(n')_2$ and $y_j(n')_3$ are simply multiplied by w to amplify their difference

values. The (OLD) wy_j values are 2,4 and 6. The adjusted (NEW) $w'y_j$ values are shown as 4,8 and 12. (Note that in general, the differences will not be equal.) Hence to initiate the $\psi(M)$ adjustment, all y(n') values are multiplied by w'. If the network is memoryless, only N computations are required to recompute all y(n'). If the network contains memory, n'N computations are required; in networks with memory, it is desirable and advantageous to select n' as small as possible.

The new value of ψ' can be computed as $\psi' = \psi - y_j w + y_j w'$ where w' is the new value of w during the interval corresponding to n' and ψ is the previous value of the pseudo correlation for the respective y_j. Note that the computation required is minimzed using this approach. If after a finite number of changes in w during interval n', one of the differences $\psi_j - \psi_k$, j and k are two rows, is still zero, a second interval can be used and the procedure to adjust w can again be applied. The important point to note here is that the upper limit on the number of computations can be estimated. The upper limit will be: No. (w') * No. (M) * No. (N), where No. (w') is the maximum number of variations in w during any time interval, No. (M) is the total number of samples and No. (N) is the number of functions computed. It will be roughly 1 plus the number of faults.

The algorithm does not guarantee a solution. If, however, one exists for the particular w selection policy, the number of computations required to find a w which gives distinguishability will usually be less than using a scheme which randomly selects an interval. Improvement in the algorithm will result if a weighting scheme is used to select the best interval to adjust, assuming two or more intervals have different values for y for the given fault conditions.

2.9.5 Detailed w Selection Algorithm: Fault Isolation

1. Fix all initial parameter values and model structure.

2. Specify $\underline{u}$ and initial conditions; also M, the number of output samples and Δt, the spacing of the samples.

3. a) Select a w function. Use $w(n\Delta t) = 1$ for $n = 0,1,2,\ldots,M$.
   b) Specify good parameter set, P^o.

4. Find model output values for M points in time.

5. Compute $\psi_j(M)\big|_{P_r^q} = \sum_{n=1}^{M} w(n)y_j(n)$. This value will correspond to the first of the fault settings for P.

6. Specify new parameter set P_r^q . Specify $\underline{u}$ and initial conditions as in 2. If all P_r^q have been specified, GOTO 7. Otherwise, GOTO 4.

7. Compute $\psi_h - \psi_i$ for all h = i where h and i correspond to a particular fault condition. Call this δ_{hi}. There will be ${}_NC_2$ of these where N = number of parameter sets. Are all $\delta_{hi} \geq k$ where k is a constant? If YES; you are finished and the w function selected in (3.) gives fault detection and isolation for the test specified. If NO, GOTO 8.

8. Specify all h, i that give $\delta_{hi} \leq k$.

9. Find the first $n' < M$ where $y_j(n', P_r^h) \neq y_j(n', P_r^i)$ for all h,i. Set a) $w(n) = w_n + \Delta w_n$

   or

   b) $w(n) = w_n \cdot n$

   or

   c) Select an appropriate weighting based on δ_{hi}

   If w has been altered ν times, find $n'' > n'$ where

$y_j(n'',P_r^h) \neq y_j(n'',P_r^i)$.

10. Compute $y_j(n'')$ for all parameter sets. Call the new interval n'.

11. Compute $\psi'(n) = \psi - w(n)\cdot y_j(n,P_r^q) + w'(n)\cdot y_j(n,P_r^q)$ where w' is specified in 9 and ψ is the previous value, ψ' is the new value.

12. If $n < M$ GOTO 7. If $n = M$, the weighting function gives no discrimination for all values of w for each interval n. The computation terminates.

Upon termination, a new method for adjusting w may be devised or M may be extended.

2.10 TRANSFER FUNCTION TECHNIQUES

The transfer function techniques for fault detection and isolation which have been proposed in the literature are based on the use of various input stimuli. The network is black-box modelled in terms of a set of two port parameters or in terms of the transfer function H(s). Implicit in most of the black-box techniques is the use of a network model for a priori test development and the ability to control the values of network parameters. In reality, the black-box concept refers to measurements allowed for a posteriori diagnosis. By comparing the response obtained from the actual system during the a posteriori stage with the response of the fault free network or a copy of the fault free network, conclusions about the system status can be made. We will discuss only two of the more important transfer function techniques due to Seshu and Waxman[127] and to

Valstar[141]. There are cases where these techniques are preferred to time domain techniques. If, for example, the network is stimulated by sinusoidal signals, transfer function techniques can be used to obtain the desired information.

2.10.0 Method 1: Break Point-Variation Detection

This method is due to Seshu and Waxman and has been mentioned previously. The method uses sinusoidal signals to stimulate a network and measure the gain function magnitude at the break points as a function of parameter variations.

Philosophy: The philosophy behind this method can be stated as follows. If the network under consideration is regarded as a linear two port, and is subject to parameter drift failures, then a) the "no failure" condition can be checked by ascertaining that the values of any four of a consistent set of two-port parameters are those specified or b) that in most cases, only one two port parameter is required for fault detection information and most of the time will give fault isolation information (according to this method). This latter statement was substantiated by an example. The method can be demonstrated by considering Figure 2.23.

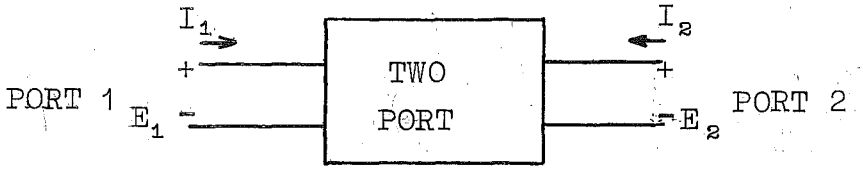


Figure 2.23

It is possible to write one of the transfer functions for the two port in the following form

$$H(s) = K \frac{(s-s_{z_1})(s-s_{z_2})(s-s_{z_3})\ldots(s-s_{z_m})}{(s-s_{p_1})(s-s_{p_2})(s-s_{p_3})\ldots(s-s_{p_n})} \cdot s^k \qquad 2.51$$

where $s = \sigma + j\omega$ and s^k, the kth order zero, has been separated out for convenience. Letting $\omega_1, \omega_2, \omega_3, \ldots, \omega_{m+n}$ be the corresponding magnitudes of s_{zi} and s_{pj} in increasing order will arrange the break points of the Bode plot for $H(j\omega)$ in increasing order. The the gain at frequencies low in comparison with ω_1 is given by

$$G_{db} = 20 \log |H(j\omega)| \cong 20 \log K + 20 \sum_{j=1}^{m+n} a_j \log \omega_j + 20 \log \omega^k \qquad 2.52$$

and $a_j = 1$ for a zero and -1 for a pole. As noted, this expression is only approximate. A shift in any of the break points or a change in K will change the low-frequency gain. To determine which break point (or K) has shifted, measurements at a frequency between break-points i and i + 1 are made. The expression for the gain at frequencies between ω_i and ω_{i+1} is given by

$$G_{db} = 20 \log K + 20 \left(\sum_{j=1}^{i} a_j \right) \log \omega_j + 20 \sum_{j=i+1}^{m+n} a_j \log \omega_j + 20 \log \omega^k. \qquad 2.53$$

Also, in general if any two breakpoints are to be tested for deviation, then by the arguments it may be seen that the upper (higher) frequency breakpoint will affect the above change in gain for both applied frequencies. This allows a comparison to be made and the breakdown responsible for the change in gain can be isolated.

The method is carried out by first writing the transfer function by the circuit (system) being

considered. Roots of the denominator and numerator are computed using standard root solving programs. A signature scale is devised which is used in the fault identification scheme. It involves assigning a number to quantized regions of the gain-change. For example, such a four signature scale might be the one shown in Table 2.1.

SIGNATURE	INTERPRETATION
0	Within .5 dB of nominal gain
1	.5 to 1.5 dB too low
2	More than 1.5 dB too low
3	.5 to 1.5 dB too high
4	More than 1.5 dB too high

Table 2.1

Next, several break points are specified. (The authors leave the reader to imagine how these might be selected for a complex system.) The following method is suggested:

a) One frequency is selected below the lowest break point.

b) One frequency is selected to be greater than the highest break point.

c) Several intermediate frequencies are selected. These should be between successive break points.

A computer program is then applied which expresses the value of the gain with all of the circuit (system) parameters set at their nominal values. Next, the parameters are varied by stepping them certain percentages plus and minus from the nominal value. For each setting,

the gain is computed at each of the selected frequencies. The signature that applies for the particular frequency is entered in a table as an element for the row and the fault list is developed as a set of signatures versus parameter setting. An example table is shown in Table 2.2.

SIGNATURE					PARAMETER SETTING
	ω_1	ω_2	ω_3	ω_4	
	0	0	1	2	R_1 + 10%
	0	1	0	4	R_2 + 10%
	1	3	1	1	C_1 + 10 %
	1	0	1	0	C_1 + 10 %
	1	0	2	0	R_1 - 10 %
	1	0	4	0	R_2 - 10 %

Table 2.2

The signature can be regarded as an encoding of the fault condition through the application of signals of the frequencies ω_1 ω_2 ω_3 and ω_4 shown in the four columns. The parameter and the amount that it is varied is given in the right hand column. This is a contrived example and of course real circuits will have much longer tables. In a 13 component circuit, Seshu obtained a table of 62 fault signatures. 41 of the signatures identified faults uniquely.

Limitations: This method has the following limitations: 1. It requires transfer functions (an assumption of linearity) with critical frequencies which are well separated. 2. For complex circuits it is not clear how frequencies are selected. 3. A model of the system must

be developed and this must correspond favourably with the actual system, otherwise correlation between actual system fault-measurement pairs with the simulation pairs will not be good. 4. It requires a wide range of sinusoidal frequencies.

2.10.1 Method 2: Modified Laplace Transform Technique

This method was developed by a group directed by J.E. Valstar [140]. It is in some respects similar to the work of Seshu in the previous section and to some work done by Moe and Murphy [105] and others [117]. Linear transformations on the input and output of a linear system are used to generate numerical information about the coefficients in the numerator and denominator of the system transfer function. The authors attempt to unify the process by considering convolution to be the fundamental operation. They claim that those methods which arise from this study which can be realized by physical filters of a simple form are very desirable because of the possibility of applying them to real time applications.

Philosophy: The objective of these methods is to generate a set of measurable voltages which can be used to determine values of the transfer function coefficients. This is done by performing transformations on the input and output which yield information on specific coefficients. It is essentially a parameter error tracking technique, hence primarily suited for fault detection.

The simplest transformation filter (also described in [141]) is the low-pass RC filter which can be regarded as performing a type of Leftsided Laplace transformation on the signals. Measurement and observational errors

seriously affect the accuracy of solutions which can be obtained by these methods. Other methods such as correlation, impulse testing and convolution are mentioned but these methods are not developed. The method can be outlined as follows.

It is assumed that the system can be described as a set of differential equations relating the input and output as

$$\sum_{m=o}^{M} p_m \frac{d^m y(t)}{dt^m} = \sum_{n=o}^{N} z_n \frac{d^n \underline{u}(t)}{dt^n} \qquad 2.54$$

where $p_o = 1$, z_o = gain at zero frequency
p's are determined by poles
z's are determined by zeros
$y(t)$ = OUTPUT
$\underline{u}(t)$ = INPUT

A transformation on Equation 2.54 is made to reduce it to an algebraic equation in the variables s and t. The transform is analogous to the Laplace Transform except that $s = \sigma$ and the integration is taken over different limits than the ordinary one-sided Laplace transform. The name that Valstar uses more often to describe the transform is the Bounded Exponential Transform although it comes under various names such as the running left-sided transform. The basic idea is that the differential equation shown in Equation 2.54 is transformed by the integral transform defined by

$$F(s,t) = \int_{t_i}^{t} f(\tau) e^{s(t-\tau)} d\tau \qquad 2.55$$

Valstar shows a method for mechanising this transform by a series of RC networks whose time constants are selected

to give various $s = -\sigma = -1/RC$. By proper choice of the time constants, it is possible to use the instantaneous information to compute the differential equation coefficent values. From these, resistance, capacitance and inductance values are computed.

For example, suppose that it is required to find the values of the parameters of a two pole, single zero network. Its differential equation is given by

$$y(t) + p_0 \frac{dy}{dt} + p_1 \frac{d^2y}{dt^2} = z_0 u(t) \qquad 2.56$$

Applying the bounded exponential transform, Equation 2.55 to Equation 2.56 gives

$$y(s_{11},t)[1+s_1p_1+s_1^2p_2] + y(t)[p_1+s_1p_2] + \frac{dy}{dt} p_2 = z_0 u(s_1,t) \qquad 2.57$$

To eliminate $\frac{dy}{dt}$, a second transform in the variable s_2 is computed giving,

$$y(s_2,s_1,t)[1+s_1p_1+s_1^2p_2] + y(s_2,t)[p_1+s_1p_2+s_2p_2] + y(t)p_2 = z_0 u(s_2,s_1,t) \qquad 2.58$$

which can be written in matrix form as

$$\begin{bmatrix} y(s_{21},s_{11},t)\,s_{11} + y(s_{21},t) & y(s_{21},s_{11},t)\,s_{11}^2 + y(s_{21},t)(s_{11}+s_{21}) + y(t) & -y(s_{21},s_{11},t) \\ y(s_{22},s_{12},t)\,s_{12} + y(s_{22},t) & y(s_{22},s_{12},t)\,s_{12}^2 + y(s_{22},t)(s_{12}+s_{22}) + y(t) & -y(s_{22},s_{12},t) \\ y(s_{23},s_{13},t)\,s_{13} + y(s_{23},t) & y(s_{23},s_{13},t)\,s_{13}^2 + y(s_{23},t)(s_{13}+s_{23}) + y(t) & -y(s_{23},s_{13},t) \end{bmatrix} \cdot \begin{bmatrix} p_1 \\ p_2 \\ z_0 \end{bmatrix} = \begin{bmatrix} -y(s_{21},s_{22},t) \\ -y(s_{22},s_{12},t) \\ -y(s_{23},s_{13},t) \end{bmatrix}$$

or generally,

$$\underline{C}\,\underline{P} = -\underline{y} \qquad 2.59$$

Mechanisation of this function requires 6 filters to obtain values of u and y transformed by $s_{11}, s_{12}, s_{13}, s_{21}, s_{22}$ and s_{23} at instants of time t. Each of the 9 terms in the matrix will be a real value at a particular time t. Valstar has shown that the parameters P can be found by inverting the C matrix in equation 2.59 (b).

Limitations: The limitations of this method are mainly the limitations in inverting large matrices and also the measurement errors occuring in the c_{ij} terms. Although the authors development of the method is mainly heuristic and several questionable assumptions are made, the method appears to work for the example given. Whether it works for practical systems or only for the example illustrated is open to further investigation.

2.11 SUMMARY AND CONCLUSION

In this chapter, three techniques have been developed for generating diagnostic information for a class of analogue systems. The class of systems considered are: all passive networks composed of resistive, capacitive, inductive and active electrical components operating in the linear mode. (We have excluded switching mode networks which were treated in Chapter 1 and which will be discussed in Chapters 3 and 4. It is conjectured that these methods can be applied to a larger class of networks than was actually treated. We have assumed that all models in this chapter are isomorphs of the physical networks. Consequently, the methods may work for homomorphic networks but examples in this chapter have assumed a one-to-one model.

Two of the methods developed are based on a scheme for fault encoding from sampled quantized output information. The third method devlops a transformation of the sampled output sequence which utilizes a weighting function. The first two methods are similar. The encoding of faults was done by varying parameters in the model to values corresponding to fault conditions. If the output signal is different from the good output, this fact is recorded. Using the cubic and planar methods, necessary and sufficient conditions for fault detection and isolation were established (using these techniques). The third method, pseudo correlation, is an alternative to the first two which need not depend on quantizing the output signal. Using discrete, evenly spaced samples, we obtained a transformation which converts the output sequence to a single number. An algorithm for generating a set of numbers, one for each fault condition was described which is useful for obtaining a priori information for fault detection and isolation.

The advantages of these methods over transfer function techniques are:

1. They use natural system signals.
2. They use information which can be obtained by practical measurement methods.
3. Their thoroughness can be assessed using the testability measure.
4. They will work for networks containing certain types of memoryless non-linearities.
5. They produce information in a form which can be readily incorporated into automatic check-out hardware.

The disadvantages of the methods are:

1) They require accurate models and facilities for computing model parameter variations for a large number of parameter values.

2) They assume that it is possible to know the initial state, the input signal and that sampling and quantization are accurate.

Applying the approach discussed here, fault information can be developed before the network is built. Fault effect analysis can then be considered as another aspect of system design evaluation.

CHAPTER 3 PAGE NO.

3. COMBINATIONAL NETWORKS

3.0 INTRODUCTION

Combinational networks are discrete signal networks whose input-output signal behaviour is described by Boolean algebraic expressions of the form[77]

$$y_i = F_{B_i}(x_1, x_2, \ldots, x_j, \ldots x_m) \qquad 3.1$$

where y_i is a single output terminal, F_{B_i} is a particular Boolean mapping or function and the x_j's are inputs. Both y_i and the x_j's are restricted to be binary values 0 (FALSE) or 1 (TRUE). The particular value of y_i will depend on F_{B_i} and on the values of the x_j's. A combinational network which realises a particular Boolean function is constructed from gates which perform logical AND, OR, and NOT operations or the equivalent.

Under normal operation (good operation) that is, when all of the gates are functioning properly, using the Boolean function it is possible to compute the output for any given combination of inputs. Or, assuming that the physical network is available, all combinations of inputs can be applied and the output for each input combination recorded. The objective of applying test inputs to a combinational network is to determine if the network is operating normally. A fault will cause abnormal operation which can be detected by applying certain input combinations and observing the resulting output signal or signals. Gate faults occur in the following forms:

1. Outputs are stuck at either 0 or 1.

2. Terminals are shorted together.

3. Input signals to individual gates are blocked

because of open or short circuited diodes or transistors or because connecting wires or leads are open circuited.

4. Output transistors are operating abnormally; there may be conditions leading to reduced current gains or biasing resistors may be out of tolerance causing transistors to unsaturate.

5. Loose terminals or improperly joined connections cause intermittent faults when the circuit board is subjected to vibration or temperature effects.

Using physical arguments, one can demonstrate that many faults which occur in logic gates will result in a "stuck-at" output signal condition. Several authors[87,120] have previously discussed the justification for assuming this faulting mode. We will assume in this chapter that all of the physical faults can be simulated by the stuck-at-zero or stuck-at-one output condition. In the FE models described in Chapter 1, it was shown that error (fault) simulation could be accomplished by providing auxiliary parameters. The model used in this chapter is the faultable gate. The faultable gate directly simulates a mapping error by fixing parameter valves appended to the good gate function. Using the faultable gate, the good operation and stuck-at-one or stuck-at-zero operation of the gate can be simulated. The latter two modes give the gate an output value independent of what the inputs might be.

The factor that makes combinational network diagnosis difficult is the non-unique input-output response. For a given input pattern, the output may be the same both for the good and faulty network. For this reason, input signals and output terminals must be selected which will

give responses which depend not only on the inputs but also on the status of the network gates. The a priori methods illustrated in this chapter are based on input signal and output terminal selection. Both fault detection and fault isolation will be discussed.

The problem of system diagnosis outlined in Chapter 1 was shown to consist of several parts, diagnostic models and test studies being particularly important. We will assume in this chapter that the system has already been partitioned and that purely combinational networks have been separated out for fault analysis. In combinational network diagnosis it is valid to assume that the input signal patterns can be selected and that any of the gate output terminal signal values for any input combination can be computed. This freedom to select enhances our ability to develop efficient fault detection and isolation tests.

We also mentioned in Chapter 1 that generation of combinational network diagnostic information has taken several forms. Prominent among the various techniques are those of Roth[122] who uses a tabular reduction technique based on his D-cube calculus and the algebraic method of Poage[115]. The method developed in this chapter for generation of diagnostic information using the faultable gate is similar to the method of Poage. The similarities, and differences between these methods and the method developed in this chapter, will be pointed out in Section 3.6. It will be shown, for example, that the faultable gate can be adapted to a simulation technique which can be used to generate diagnostic information for large single output combinational networks and for multiple-output networks.

The roles of diagnostic information generation and presentation cannot always be sharply defined. For example, Roth's method simultaneously generates and reduces the network expression by an intersection process. Network inputs which can be used to detect all single faults are produced as an "output" from the procedure. In his method, the presentation is an intergral part of the reduction process. The end product of the reduction process is the set of inputs required for fault detection testing. In a similar vein, Poage develops algebraic expressions which can be directly reduced to obtain the set of inputs useful for fault detection.

In addition to the work of Roth and Poage, there are a number of accounts on the application of test tables, decision trees, test diagrams, and tracing for developing serial fault isolation tests[23,49]. These require a larger number of input-output observations that the fixed fault detection tests (See Figure 1.18, Chapter 1). The important contributions on the use of test tables for developing combinational network fault detection and isolation test procedures are by Armstrong[7], Chang[23], and Kautz[69]. The test table and its derivatives are methods for presenting test results in a graphic form which permits a sequential fault detection or isolation test to be developed by inspection. Poage was one of the first to use fault tables as a method for developing efficient fault isolation tests and Armstrong, Chang and Kautz have refined and extended his ideas. They develop procedures which are optimal in the sense that they require the minimum number of input combinations to detect (or isolate) all possible faults.

In all diagnostic studies, and particularly for combinational networks, the method for presenting the diagnostic information is very important. Because a large quantity of information must be stored and manipulated for a posteriori fault detection and an even larger amount for fault isolation, it is necessary that the information be condensed as much as is practicable. This means for example, that long binary vectors which are used to record input and output information and fault conditions should be stored as decimal or perhaps hexidecimal numbers, the objective here being diagnostic data which can be used in automatic checkout schemes and easily interpreted by the diagnostic engineer. Because efficiency is one of the primary goals, tests which require the least storage are to be preferred.

3.0.0 Chapter Objectives

In this chapter we develop techniques for combinational network diagnosis within the framework outlined in Chapter 1. In particular, the faultable gate will be employed in two applications. The first application is in an analytical procedure for developing single output network fault detection tests which are efficient and whose effectiveness is measurable and guaranteed. The second application is to fault simulation in a computer program which computes any or all outputs in networks whose inputs can be selected independently or generated by an algorithm and in which all single faults are assumed. The algorithmic generation is suitable for specifying tests on networks too large to be handled by the analytical procedure.

We will treat the following separate problems: 1) the specification of criteria for selecting the minimum

number of input combinations for fault detection in single and multiple output combinational networks, 2) generation of diagnostic information using algebraic techniques developed from faultable gate Boolean expressions, and 3) generation and display of information for developing tests both for detection and isolation of faults in single and multiple output combinational networks.

In Section 3.2, several new test efficiency criteria are developed. This section is followed by a discussion of the primary test decision diagrams in Section 3.3. Some of these were briefly mentioned in Section 1.7 of Chapter 1.

The main contribution original to this thesis and presented in this chapter is the development of the faultable gate as a device for injecting network fault conditions analytically. As mentioned, the basic concept resembles the work of Poage but network expressions containing the fault controlling parameters are generally considerably less complex. The final expression is somewhat less complete in the sense that the faultable gate expressions will not accurately simulate conditions in networks where fan-out is greater than one.

The significant simplification in the network equation justifies this approach when there is no fan-out. However, in networks which contain fan-out, modifications may be made to the network expression which artificially handle the fan-out problem. The resulting expression is generally less complex than the equivalent Poage representation.

In the analytical adaption of the faultable gate, the Boolean equation for each network output is expressed in a sum-of-products form. This equation contains the

good and the stuck-at-one and stuck-at-zero conditions in implicit form. Affixing values to the parameters specifies a particular fault condition. The laws of Boolean algebra are applied and the resulting expression yields the inputs which must be applied to detect each particular fault.

The simulation is similar in its direct use of the faultable gate expressions. The information on the gate type and connections is fed into the simulation which generates outputs for each gate and for all one-at-a-time faults for any given input combination. The test inputs can be selected by algorithms or by the information from analytical studies using the literal expressions for the network.

The efficiency of the test types developed here can be evaluated by using the testability measure presented in Chapter 1 and in this chapter in Section 3.1.1. It is believed that this simulation represents a significant result in multiple output combinational network test information generation and display in its present form and that it can be developed into a powerful diagnostic tool by extending the size of network (increasing the memory requirements of the computer) which it can handle and by adding additional algorithmic and display options.

3.1 MEASURES OF TEST EFFICIENCY

3.1.0 Types of Tests and Limitations

Three types of a posteriori tests were defined in Chapter 1. These are the combinational or fixed, the sequential or serial and the adaptive. A fixed test on a combinational network applies a predetermined input or

series of input patterns and the outputs obtained are compared with the outputs obtained for a good copy of the network. If at any point, the two outputs disagree, a fault has been detected. A sequential or serial test uses a fixed input for the first pattern. The next input to be applied depends on the present input-output pair. This type of test is preprogrammed in the sense that all possible results are accounted for and each possibility designates the next input pattern to be applied. An adaptive test is more complex. The next test input to be applied depends on not only the input-output values at the present time but also on previous values and additional factors such as the total number of input combinations so far applied.

The tests developed in this chapter will be of the first two types. For fault detection, the fixed test will be used. In one sense, a fault detection test is a "pre-programmed" serial test. If the output for all input combinations is identical to the output for the good copy of the network, the network is judged good. For any intermediate stage, the network can only be judged possibly good. For fault isolation, the serial test is required.

The reader is reminded again that the networks are nonredundant and contain no feedback paths. The first condition precludes the possibility of inherently undetectable faults $\binom{125}{15}$. The second condition excludes the sequential networks which will be discussed in the next chapter. Although sequential networks cannot in general be treated with the techniques described in this chapter, networks with output triggers, or with output memory devices only can be handled if special routines are appended to the general algorithms.

Because a fault detection test can be shorter than a fault isolation test, it is usual first to apply a fault detection test to a particular network and if a fault is detected, a fault isolation test is then invoked. This approach was mentioned in Chapter 1, Section 1.7. If multiple outputs are prescribed, it may be possible to carry out fault detection and isolation testing concurrently. However, if only a single network output is available, the fault detection test followed by a fault isolation test is probably the most satisfactory approach*.

Whether it is optimal to use a multiple output test or a single output fixed-serial approach depends on many factors; cost of additional outputs, total number of outputs required and type of application (environment) of the network being some of the more important. The key point to be made here is that a new option is developed - the multiple output approach. It does not depend on adding new circuitry to the network but only on making certain outputs available for test points. The optimum number of test points is developed in the next sections.

3.1.1 General Test Efficiency Criteria

For digital network tests, we specify two important cost factors. These are:

1. Cost per test point.

2. Time required to perform a given test.

In this discussion, a test point will be defined as an internal network terminal, not ordinarily available for display. In special cases, internal inputs may be

---

*We are of course excluding hardware error detection circuitry (125).

selected for stimulation but a more reasonable approach is to consider using the inputs which appear as natural inputs to a package, a module board, or which form the bits of a register. The test outputs comprise the natural network output(s) and the test point outputs. The cost of the test point might be simply that of incorporating the terminal into the circuit board and into the test equipment. However, if test points are available only for manual interrogation, the inevitability of decreased checkout efficiency must be recognized as a cost.

The time required to execute a given test is roughly proportional to the number of input combinations required. If every gate output is made available as a test point, then only a few inputs will be required to determine that the network is functioning correctly. Including every gate output terminal as a test point is a very costly and perhaps impossible solution. So although minimum test time could be achieved by including every terminal in the network as an output, this is obviously going to be an unsatisfactory solution and a compromise must be made between test point numbers and test time. Because of the speed of digital units, it is possible to apply severl thousand combinations and evaluate the results in a matter of seconds. This approach is presently applied in automatic check-out schemes[31,66].

3.1.2 General Effectiveness Criteria

For a given test consisting of fixed input and output vector dimensions and fixed sequence of input combinations, the test effectiveness is measured by the testability measure. In Chapter 1, testability measure defined the effectiveness of a given test to detect all network

failures. We can modify and extend the definition to include fault isolation. Two part diagnosis results in the following definitions:

> Definition: Detectability is defined as the ratio of faults detected by a test to the total possible faults.
>
> Definition: Isolatability is the ratio of faults which can be isolated by a particular test to the total number of faults which can occur.

Both definitions will be applied to single faults. They are restricted to this condition first of all because it is realistic for combinational digital networks and second because an attempt to treat all faults would require an astronomical number of solutions.

Isolatability is highly dependent on the system organization and the component definition. The problems which have been previously treated are isolation to a sub-system, isolation to a module and isolation to a gate[23,49]. With the advent of LSI, the requirement for isolation to a single gate will undoubtedly be superseded by isolation to an integrated circuit block.

3.2 OPTIMUM TESTS

3.2.0 Fault Detection Tests

A fault detection test on a combinational network results in a decision Δ, that the network is possibly good, good or bad. It consists of applying an input combination and observing the output signals. If these signals correspond to those obtained from the good network, the network is judge possibly good, if the test is incomplete, and good, if it is complete.* If any of the outputs

* This can be determined by using a testing diagram.

differ, the network is said to be faulty or bad.

An optimum test will be one which uses the smallest subset of the 2^m possible input combinations available where m is the number of input terminals. In general, the number of combinations required will depend on the number of output terminals available and the number and type of gates in the network. The notation used in this section will represent an input combination by I and an output combination by O. The terminal sets for these are given by $\{x_j\}$ and $\{y_i\}$. The Boolean function describing the network will be given implicitly as the mapping M_B. g stands for good and f denotes a fault. An example will be used to illustrate the conditions for optimum tests. Figure 3.0 shows a four gate combinational network with three inputs and one output.

Example Gate Network

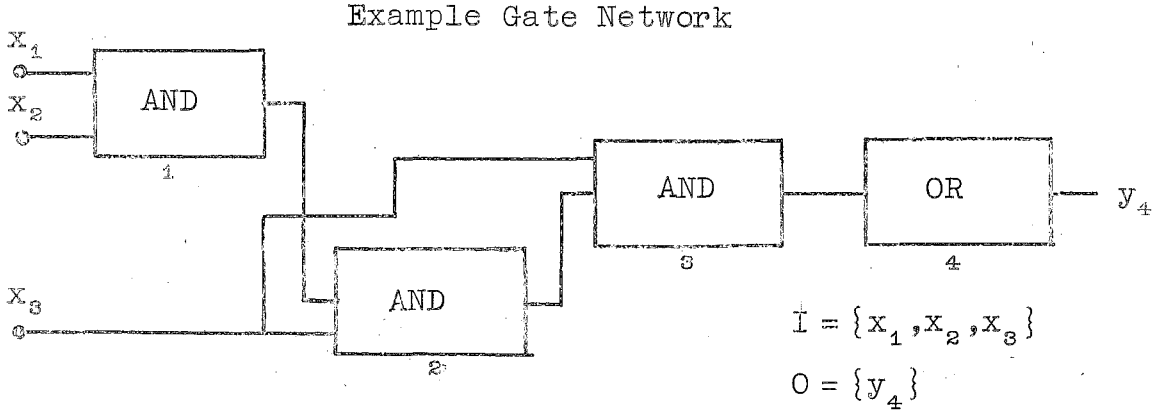


Figure 3.0

For some given input combination I_1^*, the above network gives an output O_1^g where the superscript g means the

---

* The subscript on I and O will be interpreted as either a general classification notation or later, as the decimal value of the binary input set - (the set is ordered according to some selected convention.)

network is good. In terms of the mapping M_B this is denoted by

$$M_B : I_1 \rightarrow O_1^g \quad . \qquad 3.2$$

If we assume that faults cause the gate outputs to be stuck-at-zero or at one for a certain subset of faults $\{f_i\}$, the network mapping M_B will be changed to M_B' and the mapping given in Equation 3.1 becomes

$$M_B' : I_1 \rightarrow \overline{O}_1^{\{f_i\}} \qquad 3.3$$

where the input is the same as in Equation 3.1 but the output $\overline{O}_1$ is the <u>complement</u> of the good output for the set of fault conditions denoted by f_i. For fault detection, it is assumed that a network containing n elements (gates) can have one good state and 2n faulty states assuming single faults of the stuck-at-one (s-a-1) or stuck-at-zero (s-a-0) type.

An <u>optimal fault detection</u> test consists of just two input-output pairs. The first input maps as

$$M_B : I_1 \rightarrow \overline{O}_1^{\{f_{n_1}\}}$$

where $\{f_{n_1}\}$ is the subset of faults which give the output $\overline{O}_1$. The second input-output pair gives the mapping

$$M_B : I_2 \rightarrow \overline{O}_2^{\{f_{n_2}\}}$$

where $\{f_{n_2}\}$ is the subset of faults which gives $\overline{O}_2$. The test is optimal for

$$|n_1 + n_2| = 2n \qquad 3.4$$

where n_1 and n_2 are the number of elements in the sets $\{f_{n_1}\}$ and $\{f_{n_2}\}$ respectively.

Illustration

The table in Figure 3.1 shows the fault table giving optimum conditions for the example in Figure 3.0.

x_1	x_2	x_3	g	f_1^0	f_2^0	f_3^0	f_4^0	f_1^1	f_2^1	f_3^1	f_4^1
1	1	1	0	1	1	1	0	0	0	0	1
0	0	1	1	1	1	1	0	0	0	0	1

Note that the entries are for the actual value of y_4.

Fault Table for Figure 3.0

Figure 3.1

The inputs are particular combinations of x_1, x_2, and x_3. There is a column labelled g (good) which is 0 for the input condition 111 and 1 for the input 001. The reader can verify that all faults are detected by the two inputs. It might be argued that only one input is required. For instance, a particular network might give a value 0 for the g column and output 1 for all the f columns. This is of course unrealistic because if the output gate is changed from s-a-1 to s-a-0 or vice versa, the output must certainly change value. This would mean that at least one entry in the f columns must be different from the g column. Hence, there must be at least two inputs to detect all faults in an arbitrarily complex network. The reader will appreciate that the organization of the table in Figure 3.1 is not unique in the f columns. As shown, f_1^0 - f_4^0 are s-a-0 faults and f_1^1 - f_4^1 are s-a-1 faults. The important point is that the fault associated with each f_i^j must be identified as being different from some other fault. In the next section the use of the

fault table for isolation will be demonstrated.

3.2.1 Fault Isolation Tests

3.2.1.1 Single Output Tests

As mentioned previously, the occurrence of a fault has the effect of either changing the function associated with the network in which it occurs or of altering the values of the network variables. It is possible to write a truth table for the network under each fault condition. Assuming that the function is unique for each input-output pair, it is conceivable that a particular element will respond correctly to one I - O pair but not to any others. Certainly if the network represents a canonical form, this will be the case[77].

For single output networks, the optimum number of input combinations equals the number of faults to be isolated. If the input vector has m elements, then assuming that faults occur singly and that they are of the s-a-1 or s-a-0 type, the minimum number of inputs required to isolate faults in gates for single output networks is given by

$$m \geq |\log_2(2n)| \qquad 3.5$$

The | | notation denotes the nearest integer rounded upward. The time required for a fault isolation test that is optimal would be (2^m) × (the duration of a single input combination.) In an automatic check-out scheme, the time to simulate the network, compute the output and process the results would be figured into the duration time.

3.2.1.2 Multiple Output Tests

The conditions for optimal multiple output tests can be developed through a line of reasoning similar to that used for single output networks. If ℓ outputs are selected for test points (ℓ greater than one) then there are $2^{\ell} \cdot 2^{m}$ total I-O combinations. Under ideal conditions, 2^{m} of these will be unique to the good network. Under the same conditions, there are $2^{\ell} - 2$ possible unique fault indications for each input combination. For optimal conditions, the relationship between inputs, outputs and faults is

$$2^{m}(2^{\ell} - 2) = 2n \qquad 3.6$$

For m fixed, the minimum possible number of required outputs is, from Equation 3.6,

$$\ell = 1 + |\log_2(n2^{-m} + 1)| \qquad 3.7$$

Example

Compute the minimum length output vector for a network containing 130 gates if the number of inputs m, is 7. Applying Equation 3.7 directly gives

$$\ell = 1 + |\log_2(\frac{2 \cdot 130}{2^7} + 1)|$$

$$= 1 + |1.52|$$

$$= 3 \quad .$$

There will be 1024 possible combinations using 7 inputs and 3 outputs. Of these, 128 will be the outputs for the good network and at least 260 will be unique to the particular failures. For each input combination there will be one good output, one or more outputs which

indicate a failure, and outputs which are the same for several failures. These possibilities can be observed from the following illustration.

Consider the fault table in Figure 3.2. Here again, entries are actual output values. The inputs are x_1 and x_2 and the outputs are y_1, y_2, and y_3. The fault table is developed by rows.

x_1	x_2	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0	0	001	001	000	101	001	001	101	001	101
0	1	101	100	101	010	101	000	101	101	101
1	0	111	111	111	111	100	101	111	111	110
1	1	000	000	101	000	100	101	111	010	111

Figure 3.2

For faults which give outputs identical to the good network, a single underline _____ has been used. If two or more entries in the row are identical, one or more underlines are used. In row one, f_1, f_4, f_5 and f_7 give outputs identical to the good network. Faults f_3, f_6 and f_8 give the same output, 101. Hence they have been underlined twice. The input 00 uniquely isolates f_2 (output 000) and detects faults f_3, f_6 and f_8. In summary, the tests and the faults which they isolate are:

Inputs	Faults Isolated
00	f_2
01	f_1 f_3 f_5
10	f_4 f_5 f_8
11	f_4 f_7 (f_6, f_8)

The reader will appreciate that the test is by no means optimal. Using Equation 3.6, the theoretical maximum number of faults detectable is 24, for m = 2 and m = 3. In the above summary table, the faults f_6 and f_8 give identical outputs for the input 11. In this case, it is necessary to apply input 10 to determine whether the fault is f_6 or f_8. If the fault is f_6, the input 10 will give an output 111 as shown by Figure 3.2. For the fault f_8, the input 10 gives 110 uniquely.

Isolation of faults f_6 and f_8 requires more than one I - O observation to distinguish between them. Consequently, a sequential test is required for fault isolation for the example in Figure 3.2. More will be said about sequential tests when fault tables and decision trees are discussed in the next section.

3.3 TEST INFORMATION DISPLAY

3.3.0 Introductory Comments

Two methods for displaying test result information were described in Chapter 1. These were the fault table and the testing diagram. The latter has also been termed a test tree or decision diagram. In this section, we want to show how these two essentially equivalent test display methods can be used to develop combinational network tests. Also in Chapter 1, we described various test types. The particular test type selected dictates whether an a posteriori test can be described as fixed, serial or adaptive.

Testing diagrams offer a display medium for recording the results of a series of single test decisions. For combinational networks, the interpretation of a test

decision for fault detection is simple: the output for a given input is either the same as the good output or it is different. The fact that not all faults cause the output to be different from the good output leads to the possibly good or iffy result. However, using the testing diagrams, fixed tests can be converted to serial or adaptive tests.

The test type selection procedure area delineated in Chapter 1 is important in combinational networks because of the very large number of possible combinations of faults, inputs and outputs which can be obtained. The testing diagram is useful for developing various combinational tests using the basic test decision. The objective is to apply an optimum sequence of input patterns to a specified network and in so doing, obtain the maximum amount of test information.

The ability to obtain theoretically optimum test procedures depends on several factors. The important ones are a) the number of assumed failure modes of the network components (gates), b) the size of the network being analysed, c) the amount of information required to generate failures and d) the method for displaying the test information. We now discuss these in a little more detail.

The failure modes that are selected should imitate those exhibited by the physical gates. In the Section 3.7, justification for assuming s-a-1 and s-a-0 faults will be given. It will be shown that actual gate failures are subsumed under these fault types for all but a few conditions[119].

The size of the network which can be analysed is usually fixed either by the schematic representation

(wiring diagram), by module definitions in the circuitry allocation stage or by some "natural" constraint such as correspondence between the logical equation representation and the network realization of the logical equation. In developing tests for a total system it may be necessary to make a division in the network. The sub-network (or networks) so obtained is then considered the test network. Maling and Allen[87], Manning[90] and Chang[23] have described "rules of thumb" for selecting sub-networks of tractable size from the larger networks.

The "amount of information", means roughly, the number of bits contained in the I - O pair. One method for condensing information which is used in the simulation to be described later is to code the binary vectors into their decimal equivalent. Unfortunately, 32 bits is the maximum number of bits which can be quickly converted into integer form by the computer used in the simulation. This problem was circumvented by subdividing the input and output vectors into two 32 bit sub-vectors. In this way, binary numbers longer than 32 bits can be converted.

Experience with various diagnostic aids has shown that the method for displaying the test information largely determines the usefulness of a particular technique for test generation. To be more specific, test sequences can often be generated by manipulating binary information provided by the test table. Test diagrams or decision trees derive information for their development from test tables. A slightly different approach is to use the logic equations or some equivalent form and deduce the behaviour of faulty networks by changing the gate functions to correspond to the functional change imposed by the fault. This is obviously a long and tedious method if done manually.

Tsaing and Ulrich[137] have described such an approach and Kruskal and Hart[73] have given a method for geometrically interpreting the data. Perhaps the most commonly used display method for combinational network checkout is the fault dictionary. This method has several merits which favour its use.

Before describing test information generation, the important combinational network test result display methods will be reviewed. We begin with the test table.

3.3.1 Test Tables

A test table (fault table) is a tabular representation of the results of a test pair comprising of an input-output combination. Griesmer[46], Poage[115], Roth[120], Kautz[69] and others have made use of the fault table or a variant. One version of fault table has been presented in Figure 3.3 which is a modification of the one illustrated in Chapter 1.

The basic fault table is shown in Figure 3.3. The type of fault is designated by f_j and the input vector is given either as I_d or as a binary number. The d

Inputs						Faults									
x_1	x_2	x_3	x_4,	...	,x	f_1	f_3	f_3	f_4	f_5	f_6,	...	,f_j		f_{2n}
0	0	0	0	...	0	YES	NO	NO	NO	YES	.	...	.		YES
0	0	0	0	...	1	NO	NO	YES	.	.	.	...	.		NO
.	.	.	.	...	.	.	.	.	.	.	.	...	.		.
.	.	.	.	...	.	.	.	.	.	.	.	...	.		.
1	1	1	1	...	0	.	.	.	.	.	.	...	.		.
1	1	1	1	...	1	NO	NO	.	.	.	.	...	.		NO

Basic Test Table (Fault Table)
Figure 3.3

which subscript denotes the decimal value. The m vector elements are denoted by x_i or by x_{ij} if the inputs are specified by a two dimensional subscripting. An entry from the dth row and jth colum is denoted by D_{dj} where $d = 0,1,2,\ldots,2^m - 1$, and $j = 1,2,\ldots, 2n$. D_{dj} can be YES or NO as shown in Figure 3.3. YES means that the output O_d, due to the input I_d is different for the fault free (good) and faulty (f_j) conditions. NO means the two outputs are the same. Some authors have used logical 1 and logical 0 or vice versa[19,115]. We will use this shorter notation. 1 means "the same" and 0 means "different".

Poage[115] and earlier McCluskey[86] have noted that fault tables are equivalent to the prime implicant simplification table. Hence, techniques developed for solving prime implicant problems can be extended to fault table simplification. Kautz[69] has given the most thorough treatment to this important technique.

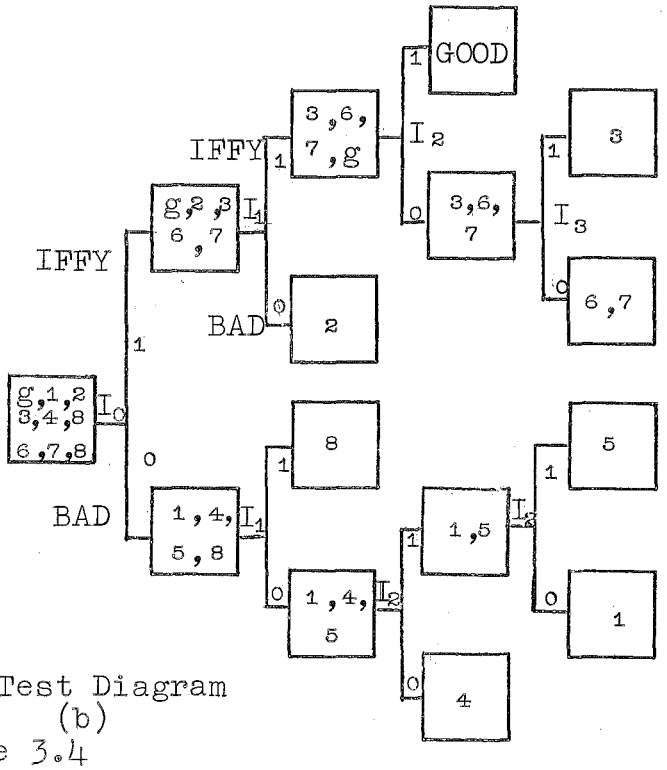
3.3.2 Test Diagrams

Test diagrams are a graphic method for reducing test information and are used for developing sequential testing procedures. An example can be used to illustrate their application to combinational network test development.

Consider first the test table shown in Figure 3.4(a). We will assume that $D_{dj} = \overline{O}_d^g \oplus O_d^{f_j}$ where $\oplus$ means addition modulo 2. If the outputs $\overline{O}_d^g$ and $O_d^{f_j}$ are the same then a 1 is entered in D_{dj}. If they are different a 0 is entered.

x_1x_2	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0 0	1	0	1	1	0	0	1	1	0
0 1	1	0	0	1	0	0	1	1	1
1 0	0	1	0	0	0	1	0	0	0
1 1	0	0	0	1	1	1	0	0	1

Test Table
(a)



Test Diagram
(b)

Figure 3.4

The corresponding test diagram is shown in Figure 3.4(b). Each box is called a branch and contains the faults differentiated by a certain I - O combination. This is equivalent to the test decision diagram in Figure 1.27 in Chapter 1. The applied input (I_d) is shown at a vertex of the diagram. The vertex has one incoming and two outgoing branches. The branching may be thought of as a set partitioning where each row set is partitioned into two subsets. To use the information for fault detection, it is sufficient to apply all I_d. If for any I_d, a $D_{dj} = 1$, the test fails and the network is judged bad.

In the test diagram, an input I_0 divides the fault set into $\mathcal{F}_1 = g, f_2, f_3, f_6, f_7$ and $\mathcal{F}_0 = f_1, f_4, f_5, f_8$. Subscripting on $\mathcal{F}$ means: a test output of 1, the network is good or faulty, and the fault is either f_2, f_3, f_6 or f_7.

If a 0 results, the network has one of the faults f_1, f_4, f_5 or f_8.

The application of I_1 further subdivides the network faults into 4 groups. Faults f_2 and f_8 are detected (and isolated) by the input I_0 followed by the input I_1. The corresponding test results are 10 and 01 respectively.

Kautz[69] has pointed out that decision trees, contact networks and test diagrams are analogous. Since they are essentially the same both in structure and application, they will not be discussed separately.

3.3.3 Fault Dictionary

The fault dictionary method for displaying fault information is essentially a more systematic version of the troubleshooting chart[57]. This diagnostic aid has been used extensively for small and relatively simple electronic equipment. The signature methods discussed in Chapter 2 are fault dictionary methods. A troubleshooting chart lists a set of voltages for the normally operating equipment and some for the abnormally operating version. If the equipment is found not to be operating properly, the voltages are compared with a set from the chart. When the sets agree, the chart indicates the probable locating of the element causing the (faulty) readings. An example of a type of troubleshooting chart is shown in Figure 3.5. There are four voltages shown; V_1, V_2, V_3 and V_4. The table is exhaustive in the sense that it contains all possible nominal, high, low combinations of the four voltages. From a knowledge of the nominal values, the actual checkout would involve a measurement of the voltages and the appropriate

classification - high, low or nominal. With the classification, one need merely to look to the REMARKS.

Key:
= Nominal
* High
- Low

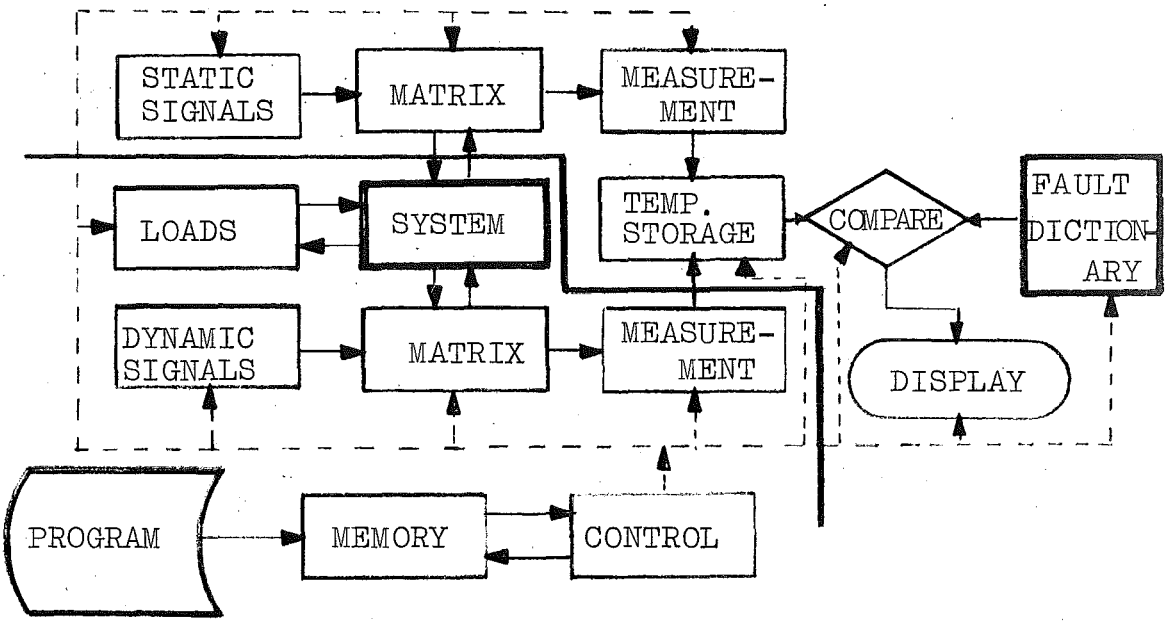
V_1	V_2	V_3	V_4	REMARKS
=	=	=	=	GOOD
*	=	=	=	R1 Low
=	*	=	=	Valve 1 - Gain
		.		
		.		
		.		
		.		
-	=	=	=	C18 or R3 Low
=	-	=	=	Transistor 13
		.		
		.		
		.		
-	-	-	-	Power rectifier, fuse

Figure 3.5

to determine the likely source of failure. This method is suitable for networks containing a small number of components. For a larger number of components such as might be found in a small digital computer, the voltages and possible faults can be listed but it is more economic to store the information on a magnetic tape or disc and retrieve the information <u>after</u> the set of measurements has been made. The likely fault(s) can then be displayed and only the minimum sorting is required, the major portion can be done automatically by a method which compares the test results with a set of stored results. When the two agree, the REMARKS pertient to the results are displayed.

3.4 THE BRIDGE FROM GENERATION TO DISPLAY

We have previously belaboured the connection between a priori and a posteriori checkout. This was done to stress the dependence of once upon the other. Figure 3.6 illustrates an automatic checkout scheme with full control over signals, loads, measurement, decisions and display;



Automatic Checkout Functional Diagram
Figure 3.6

the SYSTEM under test is outlined in bold border. We imagine the system to be a combinational network. The checkout covers both dynamic behaviour and static behaviour. To outline the confines of combinational network tests, the dark line partitioning the upper half of the diagram from the lower half is shown. The contributions from a priori combinational network diagnosis are the FAULTY DICTIONARY and the PROGRAM. These are outlined in bold border to indicate their importance here. The program will be specified by TTS, TMS and OTP. A fault dictionary is provided by TDD. In the next

section, methods for generating test information a priori for combinational networks are discussed. Some methods are useful for TDD and TTS, TMS and OTP. We will show that the faultable gate provides a mechanism for contributions in those areas. However, the distinction between display a priori for representing a TD and display a posteriori for showing an actual test decision for the equipment should be maintained.

3.5 TEST INFORMATION GENERATION

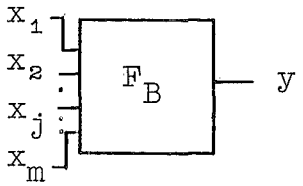
3.5.0 Fault Simulation

It was stated in the introduction that any Boolean function F_B of m variables $x_1, x_2, \ldots, x_m$ can be expressed as

$$y = F_B(x_1, x_2, x_3, \ldots, x_j, \ldots, x_m)$$

where y is a single output in this case. The x_j can have the binary values 0 or 1 and F_B is realized by a combinational gate network.

The problem of diagnosing faults in combinational networks can be made more concrete by observing that there are two possible fault sources which can exist in the network whose output is y. The schematic for the network is shown in Figure 3.7. It has a unique output for a given combination of input values. The faults which



LOGICAL NETWORK
Figure 3.7

can occur manifest themselves as (a) a change in x_j such that y for x_j is equal to y for $\bar{x}_j$ or (b) a change in the logical functioning of a network gate. These two possibilities can be illustrated by an example using a simple two input gate. We discuss the latter possibility first.

3.5.1 Single Gate Faults

(i) Functional Faults

There are 16 possible different Boolean functions F_{B_ℓ} which can be generated by a two variable. These are shown in Figure 3.8. Of the 16 shown in Figure 3.8, F_2, F_7, F_8, F_9, and F_{15} are commonly realized as gates. They

Boolean Functions of Two Variables

x_1	x_2	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 3.8

are given the names NOR, EXCLUSIVE-OR, NAND, OR and AND respectively. Logical inversion is not included in the above table but can be considered as the operation illustrated in

x	y
0	1
1	0

Figure 3.9

Figure 3.9. It is written as $y = \bar{x}$ where the bar over the top denotes logical complementation. The physical

realization is called an inverter circuit.

In general there are $2^{(2^m)}$ different logical functions of m binary variables. Any subset of functions taken from Figure 3.8 which will realize an arbitrary function of m variables is said to be functionally complete. For example, F_2 and F_9 are logical functions which form a functionally complete set. The usual test for functional completeness is to see if the function which is thought to be functionally complete can be transformed to the AND, OR and NOT (complementation) functions. Denoting the general operation by *, we can show that certain subsets of the table in Figure 3.8 are functionally complete. As an illustration, we will take F_2; $F_2 \Rightarrow *$.

Example

Show that F_{15}, F_9, and complementation can be realized by F_2 alone. The usual proof is the truth table method (Figure 3.8 is a truth table) and from Figure 3.8 we have the following:

$$\text{Complementation:} \quad \bar{x} = x * x$$
$$\text{AND operation:} \quad x_1 \cdot x_2 = (x_1 * x_1) * (x_2 * x_2)$$
$$\text{OR operation:} \quad x_1 + x_2 = (x_1 * x_2) * (x_1 * x_2)$$

These can be easily verified.

Continuing the illustration, we consider first a fault which alters the gate function. The general transformation can be denoted by

$$F_{B_k} \rightarrow F_{B_\ell}$$

where $B_k \neq B_\ell$ and $B_\ell = 1,2,3,\ldots,16$. The multi-functional property of the fault gate can be denoted by

$$y = \underset{B_\ell=}{\overset{2^{(2^m)}}{\Large E}} F_{B_\ell}(x_1, x_2, x_3, \ldots, x_m) \qquad 3.8$$

where $\Large E$ means that y is given <u>exclusively</u> by F_1, or F_2, or F_3,..., or $F_{2^{(2^m)}}$. A schematic representation for Equation 3.8 is given in Figure 3.10. This schematic is to be interpreted as follows.

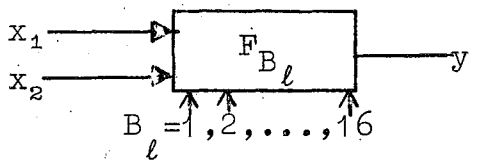


Figure 3.10

If a logical proposition is associated with each B_ℓ line, then a logical 1 on the line means that the mapping performed by the box will correspond to the Boolean function F_{B_ℓ}. Because only a single B_ℓ line may be true at any one time, the output will depend only on x_1 and x_2 and on the B_ℓ line. For the properly operating network, the B_ℓ corresponding to the function which the gate was designed to realize will be the one selected. For a gate which is faulty, the B_ℓ line will be selected which gives the desired function. In terms of the mapping error definition for a fault given in Chapter 1, a fault occurs when the mapping F_{B_ℓ} is changed to the mapping F_{B_ℓ}' where F_{B_ℓ} is the <u>good</u> mapping and F_{B_ℓ}' is the <u>bad</u> mapping.

A slightly different way of interpreting the faulty functioning is given by the sum-of-products gate shown in Figure 3.11. The input propositions and their complements are assumed to be applied to perfectly operating AND gates. The F_j blocks are more complex than they appear. They contain a block which corresponds

Sum of Products Controlled Functions

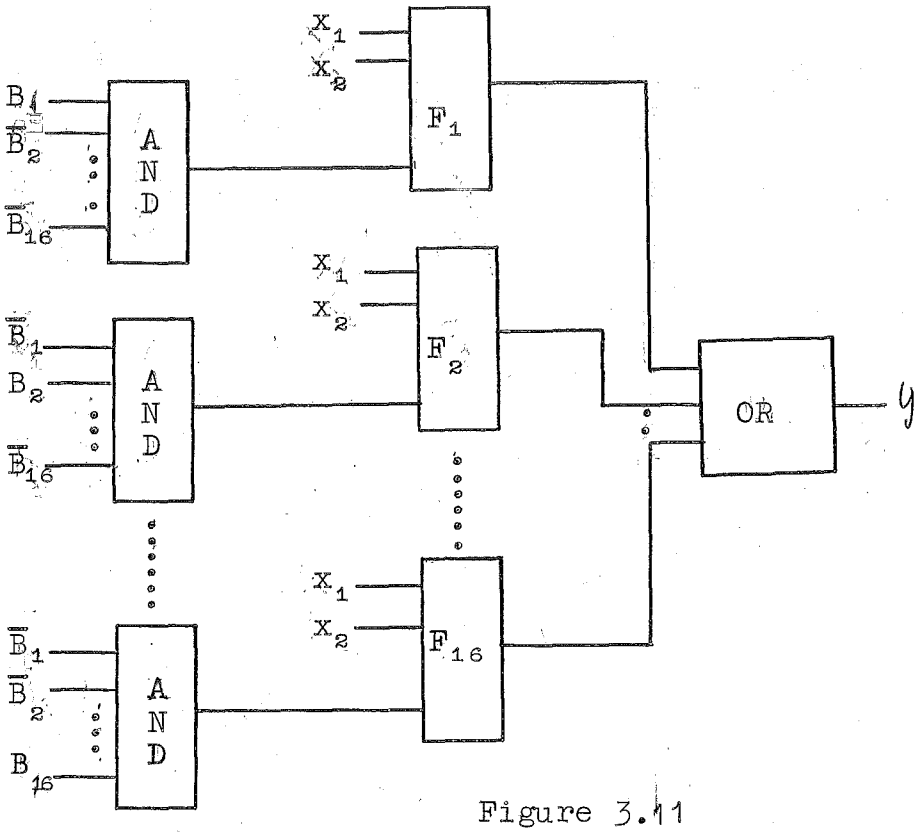


Figure 3.11

to the f_j function whose output feeds one input of a two input AND gate. The other input to the AND gate is shown at the bottom of each block. Hence, the network selects out a single f_j to apply to the OR gate. The value of y will depend on the f_j selected by the sixteen-input AND gates to which the B_ℓ are applied and on x_1 and x_2.

(ii) Line or Input Faults

Combinational tests can be regarded as single stage decision processes. Sequential tests can be thought of as multi-stage decision processes. The relationship between these can be visualized by a tree diagram. The actual processes are difficult to visualize without the aid of a display. Display methods can be used to elucidate important test alternatives to develop shortest tests provided the network to which

they are being applied is not too large.

Several types of combinational logic gate faults were mentioned in Chapter 1. A line fault is the general term applied to physical faults which block or inhibit the input signals to the gate or cause them to be permanently at one logic level. The physical manifestation might be a shorted or open circuited diode.

Large scale intergrated circuit line faults may be physically caused by the shorting together of two leads through a masking, etching or isolation layer fault. In this cause, the two input signals become indistinguishable.

Here again, it is permissible to regard these faults as a mapping error which results from a change in the network mapping. However, the effect of each line fault on the gate functioning must be computed and stored. If a fault detection test is to be designed, it may be desirable to use tests which require that every gate line be tested at both its high and low value to ascertain that the gate is good. An analytical approach to this problem has been developed by Akers[4] and more extensively by Sellers, Hsiao and Bearnson[125]. The method is to compute the Boolean difference. Basically, Equation 3.1 is used and the variable and its complement are introduced into the function. The two resulting functions are EXCLUSIVE-OR'ed and the result is 0 if the output remains the same for all input conditions. If the output is different for some condition, the result of the summation will be 1.

Suppose the Boolean difference is computed for some function with x_j and $\overline{x}_j$. Using equation 3.1 and Sellers' notation we obtain

$$dx_j = f_B(x_1, x_2, \ldots, x_j, \ldots, x_m) \oplus f_B(x_1, x_2, \ldots, \bar{x}_j, \ldots, x_m) \qquad 3.9$$

where $\oplus$ is the modulo 2 addition symbol and dx_j is the Boolean difference with respect to x_j. If dx_j is 1 for some set of inputs, then this set of inputs will make the function 1, <u>independent</u> of x_j. Using this result, it is possible to develop fault detection and isolation tests. This can be illustrated with an example.

Example (After Sellers et al)

Compute the Boolean difference to determine the conditions for which an error in the input x_1 will cause the output y to be in error for the network shown in Figure 3.12.

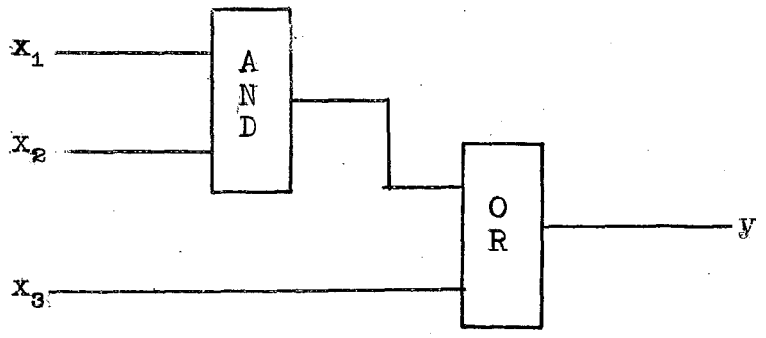


Figure 3.12

The function represented by the network in Figure 3.12 is given by

$$y = x_1 \cdot x_2 + x_3$$

and using Equation 3.9, the Boolean difference with respect to x_1 is

$$dx_1 = (x_1 x_2 + x_3) \oplus (\bar{x}_1 x_2 + x_3)$$
$$= x_2 \bar{x}_3$$

after a few lines of algebra.

Tests for input faults for each variable can be developed in a similar manner for each input. In the above example, the test input $x_2\bar{x}_3$ gives a different output for x_1 than it does for $\bar{x}_1$. This can be easily verified for Figure 3.12.

To test for all possible functional failures and all input variable failures would involve computing Equation 3.9 for each possible function (16 for the simple two-input case) for all x_j. For a large network the function becomes impossible to handle using pencil and paper. A computation for dx_j may require more than several pages for only 6 variables. Moreover, incorporating an extra 2^{2^m} gates into a network to test for all possible functional faults is unreasonable.

In seeking ways to improve upon the ideas of other investigators, and Poage and Roth in particular, it became apparent that a "mixture" of the algebraic and algorithmic approaches was probably optimum. Because the algebraic approach is potentially exact, it is useful when networks are moderately complex. But algebraic expressions become intractable when the network contains more than, say 10 gates. To handle these larger networks, an algorithm which can be programmed for the digital computer is desirable. Most algorithms are not limited by network complexity but are susceptable to computer storage size. Roth has reported[119] that a 50 gate network takes 45 seconds for one pass using the DALG algorithm and that the 360/50 model IBM which they used is limited to 100 gate networks.

The mixed approach that resulted from this work and which is described in this chapter is implemented by using the faultable gate. It is represented by a Boolean expression which can be incorporated into an

algebraic method as well as programmed for the digital computer. Using what amounts to conventional algebraic manipulations, an expression for the network output in terms of the interconnected faultable gates is obtained in a sum-of-products form. Each term in the network expression represents conditions both in terms of input combinations and gate repair status which must be present for the output to be TRUE. Using the complemented sum-of-products expansion, gate status conditions for which the network output will be FALSE can also be derived.

A purely algebraic method for test information development is difficult to program for computer solutions. Consequently, a simulation of the network must be employed when network complexity is great. The main objection to most of the previously reported simulations is that they utilize models which cannot be related to the actual network. That is, the individual gates do not exist in the simulation per se. Once the equations have been compiled in a simulator, the identity of the gates is lost. In the simulation which we have developed here, each gate signal and the gate status (good, s-a-0, or s-a-1) is known for each input to the network. This complete knowledge is invaluable when multiple output tests are required.

3.6 FAULTABLE GATE

3.6.0 Introduction

The faultable gate will be developed in this section using an example. The method for introducing faulty behaviour is to append additional logical variables to

the conventional gate network equation. A fault is simulated by selecting a particular set of values for the appended variables. Because the variables are affixed to the ordinary logical equation, the level at which fault simulation is introduced may be selected. Hence, gate faults may be simulated or an entire module fault may be simulated. This generality follows from the mapping error concept introduced in Chapter 1.

Assumptions made in developing the faultable gate are:

Assumptions:

1. Except for the primary input lines to the network, it is possible to represent an input line fault as an output line fault on the gate feeding the line. This prohibits any fan-out within the network. If fan-out exists, all of the lines being fed by the particular output will be latched to a particular logical level, (0 or 1).

2. The mode of failure exhibited by a gate is: stuck-at-one or stuck-at-zero. This corresponds to functions F_1 and F_{16} in Figure 3.8. In certain types of transistor gates, this mode of failure is dominant. Where line failures are more realistic, the reasoning in assumption 1 (above) applies. If gates are known to fail in another mode (i.e. they realize another of the functions shown in Figure 3.8), this can be accommodated by using a condensed version of the network of Figure 3.11.

The requirements for a faultable gate model are the following:

Requirements:

1. Failures simulated by the model must correspond to the failures which occur physically in the equipment.
2. Information which specifies the internal network structure should be preserved in the analytical development of the network equation using the faultable gate. (This requirement will be better understood once Section 3.6.5 has been read.)
3. The model must be suitable for incorporating into a digital computer simulation.
4. The model must be useful both for the fault free and faulty operation. In addition, it must be simple.

3.6.1 The Model

We begin the faultable gate development by observing several properties of the network shown in Figure 3.13. This network shows a main AND gate bordered by additional AND and OR gates. The main network inputs are x_1 and x_2 and the main output is y. Ancillary inputs which control the value of y are distinguished by superscripts. For the present, the network is essentially an eight input, single output network. We choose to treat it later as a single gate.

Using the network in Figure 3.13, it is possible to simulate line faults on x_1 and x_2 and stuck-at-one and stuck-at-zero faults on the output line. We can regard the inputs as being given by input and output

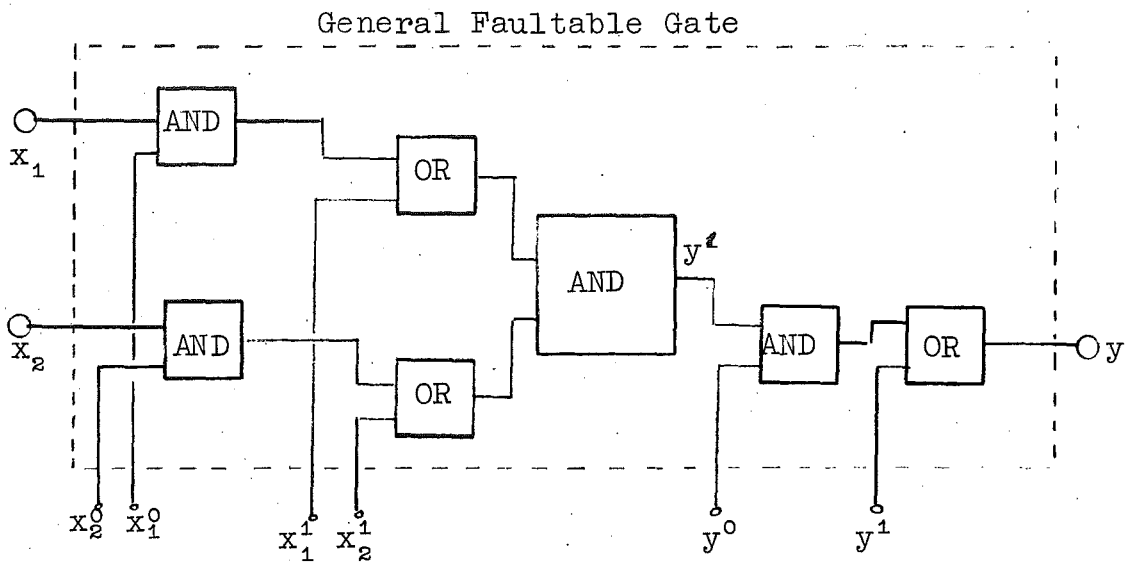


Figure 3.13

propositions which will be written

$$(x_1 x_1^0 + x_1^1) \quad \text{for } x_1$$

and

$$(x_2 x_2 + x_2^0) \text{ for } x_2 \quad \text{and} \quad (yy^0 + y^1) \text{ for } y$$

where x_i^0, x_i^1, y^0 and y^1 are termed auxiliary propositions.

Assuming positive logic, normal conditions on the input lines are prescribed by: x_1^0 and x_2^0 equal logical 1, and x_1^1 and x_2^1 equal logical 0. Under these conditions, the underlined AND gate in Figure 3.13 will be operating according to conditions prescribed in Figure 3.8. If in addition, the y propositions are y^0 equals 1 and y^1 equals zero, Figure 3.13 reduces to the gate (network) shown in Figure 3.14. This is the normal AND gate and its logical function is given by F_9 for the inputs shown

in Figure 3.8.

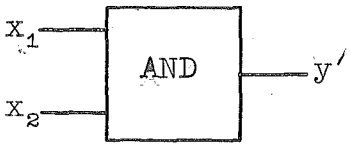


Figure 3.14

The Boolean equation expressing y in terms of x_1 and x_2 and the failure control (auxiliary) input propositions is

$$y = y^0((x_1x_1^0 + x_1^1) \cdot (x_2x_2^0 + x_2^1)) + y^1 \quad 3.10(a)$$

or expanding into a sum-of-products form

$$y = x_1x_1^0x_2x_2^0y^0 + x_1x_1^0x_2^1y^0 + x_2x_2^0x_1^1y^0 + x_1^1x_2^1y^0 + y^1 \quad 3.10(b)$$

where the "." operation sign will be implied and each clause in this expression comprises a product of logical variables termed literals. The table in Figure 3.15 shows the allowable values for the auxiliary input propositions and the resulting simulated fault. By setting the auxiliary failure propositions to the values

x_1x_2	x_1^0	x_2^0	x_1^1	x_2^1	y^0	y^1	STATE
	1	1	0	0	1	0	GOOD
	0	1	0	0	1	0	x_1 LOW
	1	0	0	0	1	0	x_2 LOW
	1	1	1	0	1	0	x_1 HIGH
	1	1	0	1	1	0	x_2 HIGH
	1	1	0	0	0	0	y LOW
	1	1	0	0	1	1	y HIGH

Table Showing Fault Simulation Conditions

Figure 3.15

given in the table, the state (gate condition) can be simulated.

Note that x_1 and x_2 columns are blank in Figure 3.15. This means that any set of inputs may be applied but the STATE over-rules any condition that may apply. If the propositions are for the GOOD state, then the network behaves as a simple two-input AND gate. Other behaviour may be deduced from the table.

3.6.1.0 Modifications

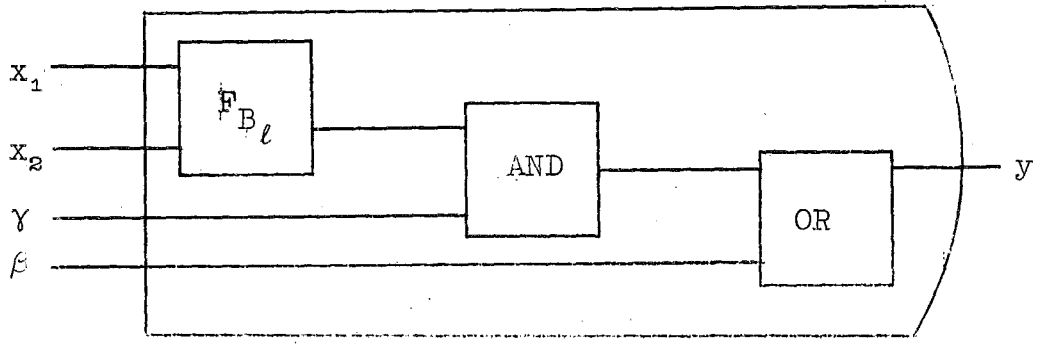
Although Equation 3.10 is basically not complex, it would be far too cumbersome and time consuming to write an equation for a network containing more than say 5 gates of the 8-input variety given in Figure 3.13. For handling larger networks, it is desirable and indeed mandatory to reduce the number of auxiliary input propositions. Poage developed a technique using what amounts to a reduction of the network in Figure 3.13 to the left of the underlined AND gate. He has not included y^0 and y^1 into his model. We are going to take the opposite side here. x_1^1, x_1^0, x_2^1 and x_2^0 will be eliminated from the model for the purposes of simplification. The motivation and reasons for introducing this simplification were given under the assumptions at the beginning of this section. We again repeat that the likely mode of failure in a gate will be either in the input line (a short or open circuit) or in a latch-up at the output. If line failures occur in the internal gates - that is, any of those gates with inputs from other network gates only - an output latch-up will be indistinguishable from a line failure if the output has no fan-out. Another

possibility is that the failure causes the gate function to change from F_j to some other function but <u>not</u> F_1 or F_{16}. This possibility can certainly be accommodated but the corresponding faultable model becomes increasingly complicated and the extra flexibility must be paid for by increasing the number of variables in the gate equation.

As a compromise between a complex gate which has the potential to simulate all 16 possible functions and the gate which can realize only a single function, a final model was selected which simulates the good and stuck-at cases.

3.6.1.1 Final Model

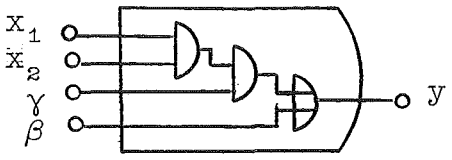
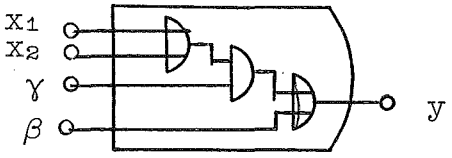
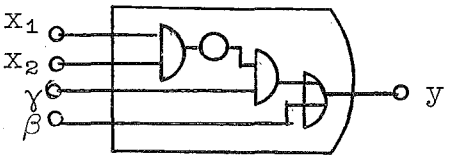
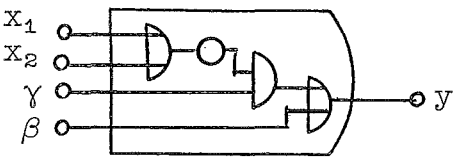
The simplified modified final model for the faultable gate in Figure 3.16. The notation has been altered slightly for the sake of clarity and for the eventual computer programming (coding). γ has replaced y^0 and β



Faultable Gate
Figure 3.16

y^1. Any function can be inserted into the block labelled F_{B_ℓ}. The remainder of the faultable gate is the same for all F_{B_ℓ}. Four of the more important faultable gates and their equations are shown in Figure 3.17. The usual designation (AND, OR, NOT etc.) can be prefixed with the capital F to distinguish the ordinary type of gate from the faultable gate. They are manipulated in the same way

as ordinary logical elements except that the γ and β are always carried as terms in the Boolean expansion.

EQUATION	REALIZATION
$y = \beta + \gamma.(x_1.x_2)$ FAND	
$y = \beta + \gamma.(x_1+x_2)$ FOR	
$y = \beta + \gamma.(\bar{x}_1+\bar{x}_2)$ FNAND	
$y = \beta + \gamma.(\bar{x}_1.\bar{x}_2)$ FNOR	

KEY: + Logical "Inclusive OR"

. Logical "AND"

- Logical Negation

Figure 3.17

3.6.2 Faultable Gate Equations

The equation applicable to each particular faultable gate is shown in Figure 3.17. An important feature of the equations is that the form of each is the same with respect to γ and β. In general, the equation for

the gate shown in Figure 3.16 can be written

$$y = \beta + \gamma(F_{B_\ell}) \qquad 3.11$$

or

$$y_j = \beta_j + \gamma_j(F_{B_\ell}).$$

where F_{B_ℓ} can be any arbitrary Boolean function. If Equation 3.11 is regarded as the actual gate equation, then the proposition associated with γ and β control the "status" of F_{B_ℓ}. The possible conditions are:

1. F_{B_ℓ}, GOOD $\quad \beta = 0, \quad \gamma = 1$
2. F_{B_ℓ}, STUCK-AT-ONE $\quad \beta = 1, \quad \gamma = 0, \ (1)$
3. F_{B_ℓ}, STUCK-AT-ZERO $\quad \beta = 0, \quad \gamma = 0$.

From the above conditions, we can see that F_B need never be altered to simulate a gate fault; the effect can be obtained by the control propositions γ and β associated with the gate model. The interpretation of a fault as a mapping error can be given if γ and β are regarded as parameters of a Boolean mapping M_B. The mappings can be written as

$$M_B(\overline{\beta},\gamma) \Rightarrow F_{B_\ell}$$

$$M_B(\beta,\gamma \text{ or } \overline{\gamma}) \Rightarrow F_{16}$$

$$M_B(\overline{\beta},\overline{\gamma}) \Rightarrow F_1 \quad .$$

3.6.3 Interpretation of Faults as Mapping Errors

One of the basic postulates stated initially was that a fault can be represented as a mapping error. This means that parameters have the effect of changing the mapping, or the Boolean function F_B depending on what

values they have been assigned.

In the previous section, we introduced the parameters γ and β into the Boolean expressions. By doing this, we made it possible to control the logical function or the value of the functions. Although the result of introducing γ and β was to "condition" the function to make it dependent on the values of the parameter settings, the result may be regarded as equivalent to a change in the Boolean mapping or simply a mapping error. Using the notation in Chapter 1 in Figure 1.3, we show the dependency by $F_B|_{\gamma,\beta}$.

3.6.4 Applications

The faultable gate has two primary applications. The first is in analytical method for specifying test input signals which will detect and in some cases, isolate failures. The method applies a reduction procedure which also specifies those faults, which are non-isolatable, hence independent of the input signals. The second application of the faultable gate is to the development of a computer fault simulation program. Here, the gate equations are themselves employed directly rather than using an indirect compiler method as is sometimes done. This simulation can be applied during the early design stages when the logic equations are being formulated or after the design has been fixed. In either case, the testability of the network for a particular test type can be evaluated.

In Equation 3.11, F_{B_ℓ} does not imply any limitations on gate complexity. It may have a simple or extremely complex representation. This means that F_{B_ℓ} may be the expression for a simple gate <u>or</u> for an entire network.

The key point is that the level of the network test can be very fine or very coarse, depending on the particular requirements. Single gates may be tested or tests may be developed at the module level.

3.6.5 Test Generation Using Analytical Faultable Gate Expressions

The application of the faultable gate expressions to generating test inputs for a given network can be illustrated by the example network shown in Figure 3.18. This has previously been used by Poage to demonstrate his technique and contains sufficient complexity. We will assume that the input terminals and output terminals are fixed and that input and output signals are binary 1 or 0. The objective of this development is to select input combinations which will give output signals which depend not only on the value of the inputs but also on whether the gates are good or contain faults.

The network in Figure 3.18 contains two AND gates, (1,2) two INHIBIT gates, 3 and 4 and, one OR gate number 5. Note that it is necessary to assign each gate a different number and that the output signal subscripts be the same

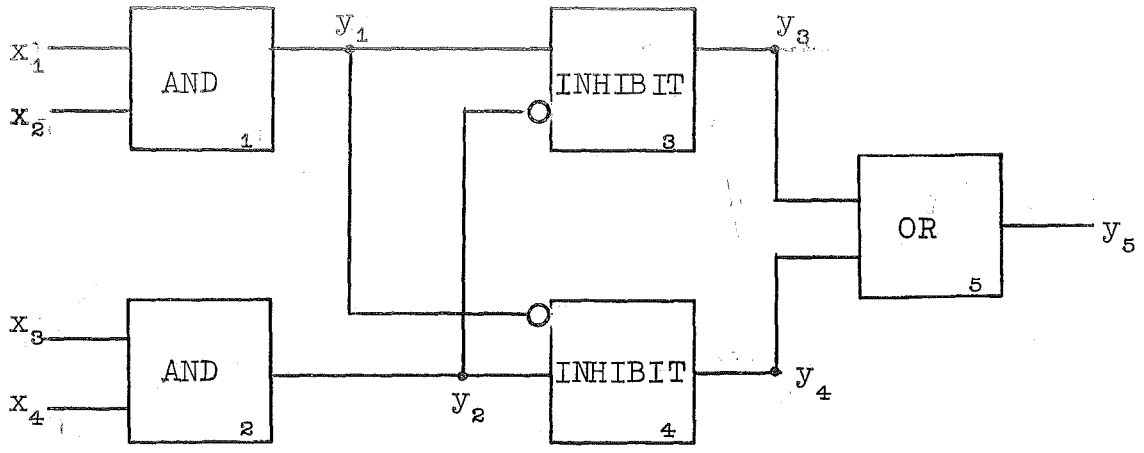


Figure 3.18

as the gate number. The γ and β variables are also assigned the subscript which corresponds to the gate number. The gate output is denoted y_i. The independent inputs are x_j. The set of faultable gate equations for each gate in the network in Figure 3.18 can be written by inspection. They are

$$\begin{aligned} y_1 &= x_1 x_2 \gamma_1 + \beta_1 \\ y_2 &= x_3 x_4 \gamma_2 + \beta_2 \\ y_3 &= y_1 \bar{y}_2 \gamma_3 + \beta_3 \\ y_4 &= \bar{y}_1 y_2 \gamma_4 + \beta_4 \\ y_5 &= (y_3 + y_4)\gamma_5 + \beta_5 \end{aligned} \qquad 3.12$$

Combining Equations 3.12 to solve for y_5 alone gives (after 15 minutes of algebra):

$$\begin{aligned} y_5 = {} & x_1 x_2 \bar{x}_3 \gamma_1 \gamma_3 \gamma_5 \bar{\beta}_2 + \bar{x}_3 \gamma_3 \gamma_5 \beta_1 \bar{\beta}_2 + \bar{x}_4 \gamma_3 \gamma_5 \beta_1 \bar{\beta}_2 \\ & + x_1 x_2 \bar{x}_4 \gamma_1 \gamma_3 \gamma_5 \bar{\beta}_2 + x_1 x_2 \gamma_1 \bar{\gamma}_2 \gamma_3 \gamma_5 \bar{\beta}_2 \\ & + \bar{x}_1 x_3 x_4 \gamma_2 \gamma_4 \gamma_5 \bar{\beta}_1 + \bar{x}_1 \gamma_4 \gamma_5 \bar{\beta}_1 \beta_2 \\ & + \bar{x}_2 x_3 x_4 \gamma_2 \gamma_4 \gamma_5 \bar{\beta}_1 + \bar{x}_2 \gamma_4 \gamma_5 \bar{\beta}_1 \beta_2 \\ & + x_3 x_4 \bar{\gamma}_1 \gamma_2 \gamma_4 \gamma_5 \beta_1 \\ & + \bar{\gamma}_1 \gamma_4 \gamma_5 \bar{\beta}_1 \beta_2 + \bar{\gamma}_2 \gamma_3 \gamma_5 \beta_1 \bar{\beta}_2 + \gamma_5 \beta_3 + \gamma_5 \beta_4 + \beta_5 \end{aligned} \qquad 3.13$$

Equation 3.13 is the sum-of-products expression for the network output. Each clause gives the conditions for which the output will be one. If all of the terms in any <u>one</u> clause are 1, the output will be one, independent of any other conditions that may exist. If at least one literal in each clause is 0, the output will be zero.

An interesting and potentially useful by-product of the expansion of Equation 3.13 is the appearance of

path terms which trace the route of signals through the network. For example, the term $x_1x_2\bar{x}_3\gamma_1\gamma_3\gamma_5\bar{\beta}_2$ shows that a 1 will appear at the output if gates 1,3 and 5 are either good or stuck-at-one and if gate 2 is good or stuck-at-zero. The input is 1101 or 1100. The actual path is through gates 1,3 and 4 and 2,3 and 5. Paths evolve naturally in the equation reduction and are equivalent to the normal form expressions described by Armstrong[7] but are developed without resorting to tracing through the network. The input combination to test the fault conditions in the expression $x_1x_2\bar{x}_3\gamma_1\gamma_3\gamma_5\bar{\beta}_2$ is 110x where the inputs are ordered according to their subscripting. The "x" denotes the fact that x_4 may be 1 or 0. We will sometimes write $\underline{x}^d$ to denote the decimal equivalent of a vector such as $x_1x_2x_3x_4$ if all members of the input set are specified. For example, the input $x_1x_2x_3x_4 = 1101$ may be written as 13 or $\underline{x}^{13}$.

For long expressions in x,γ, and β it is convenient to use the condensed notation which carries along the subscripts only. This notation is similar to the compressed notation used by Poage[115]. In this case, however, the correspondence between the two notations is more direct than Poages'. For example, the first term in Equations 3.13 can be written

$$x_1x_2\bar{x}_3\gamma_1\gamma_3\gamma_5\bar{\beta}_2 \equiv x_{123}\gamma_{135}\beta_{\bar{2}}$$

which makes the expression compact without destroying any of the useful information. Fault detection tests can be developed by using the expression given in Equation 3.13 and its complement.

Figure 3.19 is a table which shows the condensed equation for the output y_5 and its complement, $\bar{y}^5$. The

for developing all tests which detect stuck-at-one and stuck-at-zero faults using the information in Figure 3.19 will now be described.

3.6.6 Stuck-at Tests

Tests for s-a-0 can be developed by first noting that the literals in any single clause in the y_5 proposition in Figure 3.19 must all be true for the clause to give a value $y_5 = 1$. However, the clauses contain conditions for which the output will be 1 if the gates are other than good. We can eliminate all of the abnormal conditions by setting the β_i terms equal to zero. This excludes any terms which are 1 because of a possible stuck-at-one condition. A set of terms remains called the <u>reduced output proposition</u> which will give an output of 1 only if the conditions existing within the network correspond to the terms remaining. The objective is to next select an input combination which causes a clause to be 1 while all other clauses in the output expression are zero *. Then, if this input is applied, faults can be simulated by fixing the remaining γ and $\overline{\beta}$ variables to values which cause the output to be zero. A similar approach can be applied to the complemented expression to derive the stuck-at-one condition. The test development technique will be described for the equations in Figure 3.19.

Setting all β terms in the y_5 proposition false and $\overline{\gamma}$ terms in $\overline{y}_5$ false gives the resulting equations shown in Figure 3.20. Tests for s-a-0 are obtained from the y_5 proposition in Figure 3.20 by selecting sufficient γ terms to test all gates. A sufficient set of γ terms is

---

* The reader will note the similarity here to the prime implicant problem.

OUTPUT PROPOSITION y_5	OUTPUT PROPOSITION $\bar{y}_5$
$y_5 = x_{123}\gamma_{135}\beta_2$ (1)	$\bar{y}_5 = x_{1234}\gamma_{12}\beta_{345}$ (1)
$+ x_3\gamma_{35}\beta_{12}$ (2)	$+ x_{12}\gamma_1\beta_{2345}$ (2)
$+ x_4\gamma_{35}\beta_{12}$ (3)	$+ x_{12}\gamma_{13}\beta_{345}$ (3)
$+ x_{124}\gamma_{135}\beta_2$ (4)	$+ x_{34}\gamma_2\beta_{1345}$ (4)
$+ x_{12}\gamma_{123}\beta_5$ (5)	$+ x_{13}\beta_{12345}$ (5)
$+ x_{134}\gamma_{245}\beta_1$ (6)	$+ x_{23}\beta_{12345}$ (6)
$+ x_1\gamma_{45}\beta_{12}$ (7)	$+ x_3\gamma_1\beta_{12345}$ (7)
$+ x_{234}\gamma_{245}\beta_1$ (8)	$+ x_3\gamma_3\beta_{2345}$ (8)
$+ x_2\gamma_{45}\beta_{12}$ (9)	$+ x_{14}\beta_{12345}$ (9)
$+ x_{34}\gamma_{12345}\beta_1$ (10)	$+ x_{24}\beta_{12345}$ (10)
$+ \gamma_{145}\beta_{12}$ (11)	$+ x_4\gamma_3\beta_{2345}$ (11)
$+ \gamma_{235}\beta_{12}$ (12)	$+ x_4\gamma_1\beta_{12345}$ (12)
$+ \gamma_5\beta_3$ (13)	$+ x_1\gamma_2\beta_{12345}$ (13)
$+ \gamma_5\beta_4$ (14)	$+ x_2\gamma_2\beta_{12345}$ (14)
$+ \beta_5$ (15)	$+ x_1\gamma_4\beta_{1345}$ (15)
	$+ x_2\gamma_4\beta_{1345}$ (16)
	$+ x_{34}\gamma_{24}\beta_{345}$ (17)
	$+ \beta_{12345}$ (18)
	$+ \gamma_{13}\beta_{1345}$ (19)
	$+ \gamma_{12}\beta_{12345}$ (20)
	$+ \gamma_{23}\beta_{2345}$ (21)
	$+ \gamma_{34}\beta_{345}$ (22)
	$+ \gamma_4\beta_{2345}$ (23)
	$+ \gamma_5\beta_5$ (24)
	$+ \gamma_3\beta_{1345}$ (25)

Figure 3.19

REDUCED OUTPUT PROPOSITIONS

OUTPUT PROPOSITION $\bar{y}_5$		OUTPUT PROPOSITION $\overline{\bar{y}}_5$	
$y = x_{12\bar{3}}\gamma_{135}\beta_{\bar{2}}$	(1)	$y = x_{123\bar{4}}\gamma_{12}\beta_{\bar{3}\bar{4}\bar{5}}$	(1)
$+ x_{12\bar{4}}\gamma_{135}\beta_{\bar{2}}$	(2)	$+ x_{12}\gamma_{1}\beta_{2\bar{3}\bar{4}\bar{5}}$	(2)
$+ x_{12}\gamma_{1\bar{2}\bar{3}5}\beta_{\bar{2}}$	(3)	$+ x_{34}\gamma_{2}\beta_{1\bar{3}\bar{4}\bar{5}}$	(3)
$+ x_{\bar{1}34}\gamma_{245}\beta_{\bar{1}}$	(4)	$+ x_{\bar{1}\bar{3}}\beta_{\bar{1}\bar{2}\bar{3}\bar{4}\bar{5}}$	(4)
$+ x_{\bar{2}34}\gamma_{245}\beta_{1}$	(5)	$+ x_{\bar{2}\bar{3}}\beta_{\bar{1}\bar{2}\bar{3}\bar{4}\bar{5}}$	(5)
$+ x_{34}\gamma_{\bar{1}245}\beta_{\bar{1}}$	(6)	$+ x_{\bar{1}\bar{4}}\beta_{\bar{1}\bar{2}\bar{3}\bar{4}\bar{5}}$	(6)
		$+ x_{\bar{2}\bar{4}}\beta_{\bar{1}\bar{2}\bar{3}\bar{4}\bar{5}}$	(7)
		$+ (\beta_{12\bar{3}\bar{4}\bar{5}})$	(8)

Figure 3.20

a set which <u>covers</u> energy gate. For example, the terms $x_{128}\gamma_{135}\beta_{\bar{2}}$ and $x_{34}\gamma_{\bar{1}245}\beta_{\bar{1}}$ cover all gates. The first term covers gates 1,3 and 5, γ must be 1 for the test to pass, and the second covers 1,2,4 and 5. However, because the term $\gamma_{\bar{1}}\beta_{\bar{1}}$ occurs in the second clause, the stuck-at-zero condition for gate 1 must exist for this clause to be 1 if x_{34} and γ_{245} are true. A better term to select from Figure 3.20 is $x_{\bar{1}34}\gamma_{245}\beta_{\bar{1}}$. Here gate 1 must be good (or <u>possibly</u> s-a-0). The difference in these two clauses is the specification of x_1 in the latter case whereas, no specification of x_1 is made in the former. Using the two inputs prescribed by these clauses, we obtain the following results *.

x_1	x_2	x_3	x_4	g	y_1^0	y_2^0	y_3^0	y_4^0	y_5^0
1	1	0	x	1	0		0		0
0	x	1	1	1		0		0	0

Figure 3.21

Figure 3.21 shows that the output will be 1 if the network is good, and 0 for s-a-0 faults on gates 1,3 and 5 for the 110x input and 0 for s-a-0 faults on gates 2,4 and 5 for the 0x11 input. In addition, the input 110x will give a 0 output if gate 2 is s-a-1 and the input 0x11 will give a 0 output if gate 1 is s-a-1. These are not shown in Figure 3.21. The notation y_j^0 means "gate j is stuck-at-zero" and is different from the y_i^0 in Figure 1.13.

In a similar way, the reduced output proposition

*The notation in the table uses a superscript on the y to signify the stuck-at condition for the particular output. The entry in the table gives the output value for the fault condition. No entry indicates that the output is the same value as when the network is good.

for $\overline{y}_5$ can be used to generate tests for s-a-1. It is possible that different combinations result in tests on the same gates for identical fault conditions. For example in the reduced output proposition table in Figure 3.20, the y_5 clauses which may be regarded as equivalent are

(1) (2), and (4) (5) ,

and for $\overline{y}_5$

(5) (6) (7)

where the bracketed terms are those shown in Figure 3.20.

Unless there is a preference for a particular input combination or good reasons are known for retaining all terms, it is useful to simplify the Figure 3.20 to retain necessary terms only. One possible minimum-reduced output proposition table is shown in Figure 3.22.

MINIMUM REDUCED OUTPUT PROPOSITIONS

OUTPUT PROPOSITION y_5 .	OUTPUT PROPOSITION $\overline{y}_5$
$y_5 = x_{123\overline{3}} \cdot \gamma_{135} \cdot \beta_{\overline{2}}$	$\overline{y}_5 = x_{1234} \cdot \gamma_{12} \cdot \beta_{\overline{345}}$
$+ x_{\overline{1}34} \cdot \gamma_{245} \cdot \beta_{\overline{1}}$	$+ x_{12} \cdot \gamma_{1} \cdot \beta_{\overline{2345}}$
$+ x_{12} \cdot \gamma_{1\overline{2}35} \cdot \beta_{\overline{2}}$	$+ x_{34} \cdot \gamma_{2} \cdot \beta_{\overline{1345}}$
$+ x_{34} \cdot \gamma_{\overline{1}245} \cdot \beta_{\overline{1}}$	$+ x_{\overline{13}} \cdot \beta_{\overline{12345}}$

Figure 3.22

3.6.7 Fault Table Development

A comprehensive fault table can be developed using the minimum reduced output propositions. Fixed tests to detect the presence of all single faults can be developed by specifying the input combinations and the output. The fault table developed from Figure 3.22 is shown in Figure 3.23. It contains entries which give the output signal value. These are <u>not</u> the D_{ij} introduced in Section 3.3.1.

x_1	x_2	x_3	x_4	G	y_1^0	y_2^0	y_3^0	y_4^0	y_5^0	y_1^1	y_2^1	y_3^1	y_4^1	y_5^1	
1	1	0	x	1	0		0		0		0				✓
0	x	1	1	1		0		0	0	0					✓
1	1	1	1	0	1	1						1	1	1	✓
1	1	x	x	0	1	1						1	1	1	
x	x	1	1	0	1	1						1	1	1	
0	x	0	x	0						1	1	1	1	1	

Figure 3.23

We note first of all that in the fault table shown in Figure 3.23, the last two clauses in y_5 of Figure 3.22 are eliminated. The reason is that a s-a-0 condition $\bar{\gamma}\,\bar{\beta}$ must be present for the output to be good, <u>independent</u> of what the inputs may be. This brings about the conclusion that the first term $x_{123}\bar{\gamma}_{135}\beta_{\bar{2}}$ covers the third term in y_5 and the second covers the fourth term. A test which detects all single faults and which is, in this case a minimum set, is denoted by ✓ marks at the right edge of Figure 3.23.

One possible minimum fixed test given by the table is

1	1	0	x
0	x	1	1
1	1	1	1

or

1	1	0	1
0	1	1	1
1	1	1	1

1 or 0 may be inserted in the x or "don't care" entry.

It should be mentioned that Detectability is 1 for the tests derived assuming single faults.

3.6.8 Indistinguishable and Multiple Faults

Figure 3.19 equations were reduced to obtain a compact method for determining single fault detection tests. A less specific interpretation of the expression for y_5 and $\bar{y}_5$ permits the conditions for indistinguishable and multiple faults to be investigated.

For an example of indistinguishable faults, note that if the conditions indicated by clauses (11)-(15) in y_5 of Figure 3.19 prevail, the output will be 1 regardless of what inputs are applied. We can identify in these terms, multiple and indistinguishable faults. In term (11) the conditions show gate 1 s-a-0, gate 2 s-a-1 and gates 4 and 5 good. Referring to the original network, it is obvious that the inputs will have no effect on the input for this set of multiple fault conditions. Similarly, if, as term (13) of y_5 in Figure 3.19 shows, gate 3 is s-a-1 and gate 5 is good, this single fault condition cannot be distinguished from condition (11), (12), (14) or (15).

An illustration of a multiple fault which can be

detected is number (9) of y_5 in Figure 3.19. Here, gates 4 and 5 are good and gate 2 is s-a-1. If gate 1 is good or s-a-0, the output will be 1. If, however, gate 1 is s-a-1, the output will be zero.

Hence $\bar{x}_3$ tests the multiple failure gate 1, s-a-1 and gate 2 s-a-1.

3.7 TEST INFORMATION GENERATION: A SIMULATION

3.7.0 Computer Aided Diagnostic Studies

Computer simulation of logic networks has been described[89]. It provides a fast and efficient method for analysing the input-output response of a logic network. The purpose of the computer simulation program described in this section is to generate diagnostic information for a logic network. The method employed is a direct coding of the faultable gates developed in the previous section. Procedures will be described which can be used to methodically vary input combinations and simulate faults. We will limit faults to single stuck-at types. With very little modification, the simulation will accommodate multiple faults.

The simulation is oriented to testing gate networks. In the past, computing system diagnostics have been designed to test computer instructions[31]. This is an unreliable method and the results cannot be easily evaluated to obtain a testability measure. The method mentioned previously which was invented by Roth is aimed at testing the gate operations. The level tested here is the FE. Because Roth's basic algorithm depends on having single output networks, his method is not

effective for multiple output tests. The simulation described here does not suffer from this limitation. In addition, it has attractive display options.

In its present form, the simulation is "off-line". Preprogrammed input signal-fault options are specified by control cards entered through the card reader. The main options are specified as follows:

Input Signal	Fault Condition
Sequence the input vector bits to change one at a time.	Good and all possible s-a-1 and s-a-0 faults.
Read the input vector from cards.	"

Any future development should consider the possibility of using an algorithm to select the inputs on the basis of path signals[7] interrupted by a particular gate fault. Better still, a combination of algorithms and direct man-machine interaction would provide an even more useful diagnostic tool. The possibility of an on-line interaction was considered but because the IBM 360/44 system for which the program has been written is not suitable for on-line work, the pre-programmed options were used.

3.7.1 Basic Requirements, Assumptions and Applications

The basic requirements of the computer oriented fault simulation method can be states as follows:

1. The simulation must be able to generate the network equation including all outputs from a specification of the gate types and the topology.

2. The display methods must apply to multiple output networks. This means that fault tables must be in terms of all network outputs.

3. Any fault or combination of faults must be accommodated by the program provided they are the stuck-at type. The good network must be one of the possible alternatives.

4. Because single output, m input networks have 2^m possible input combinations, short-cut assumptions must be made to reduce the number of inputs required for any set of tests.

5. The simulation must be sufficiently flexible to accommodate any of the foreseeable technology. Technology alludes to hardware of the combinational type.

The above requirements and capabilities form a basic specification for the design of the simulation. The following assumptions and limitations apply to the application of the specified diagnostic program.

1. The fault types are assumed to be 'stuck-at' and may be single or multiple. In general the simulation will generate the single faults automatically. Multiple faults will be generated and tests developed only after the type required has been specified.

2. The type of faults simulated may not adequately model networks with internal fan-outs greater than 1. It is possible to circumvent this limitation by incorporating additional gates which control the fault condition.

3. Faults on input lines to gates are not tested for the independent inputs. These faults can be checked by hardware schemes such as described by Sellers et al.

The fault simulation program developed can be applied to the following general problems.

1. A priori generation of test types which are efficient and whose effectiveness can be evaluated.
2. A priori generation of diagnostic information. This may take the form of a fault dictionary or a listing of the efficient fault detection tests and the output value if the network is good.
3. The simulation without the fault option can be used to develop logic networks during the initial design stages.

3.7.2 Network Fault Simulation Program Description

General Program Structure

A block diagram of the computer program is shown in Figure 3.24. It contains four main routines plus the control program. The routines shown in the figure are READER, SUMER, LOGEQ, and OUTPUT. The basic function of the routines can be described as follows:

READER:

All of the data on the network structure, inputs, type of gates, and control variables are read under direction from this program. (55 Fortran Instructions)

CONTROL:

The control program phases the subprograms and

Simulation Block Diagram

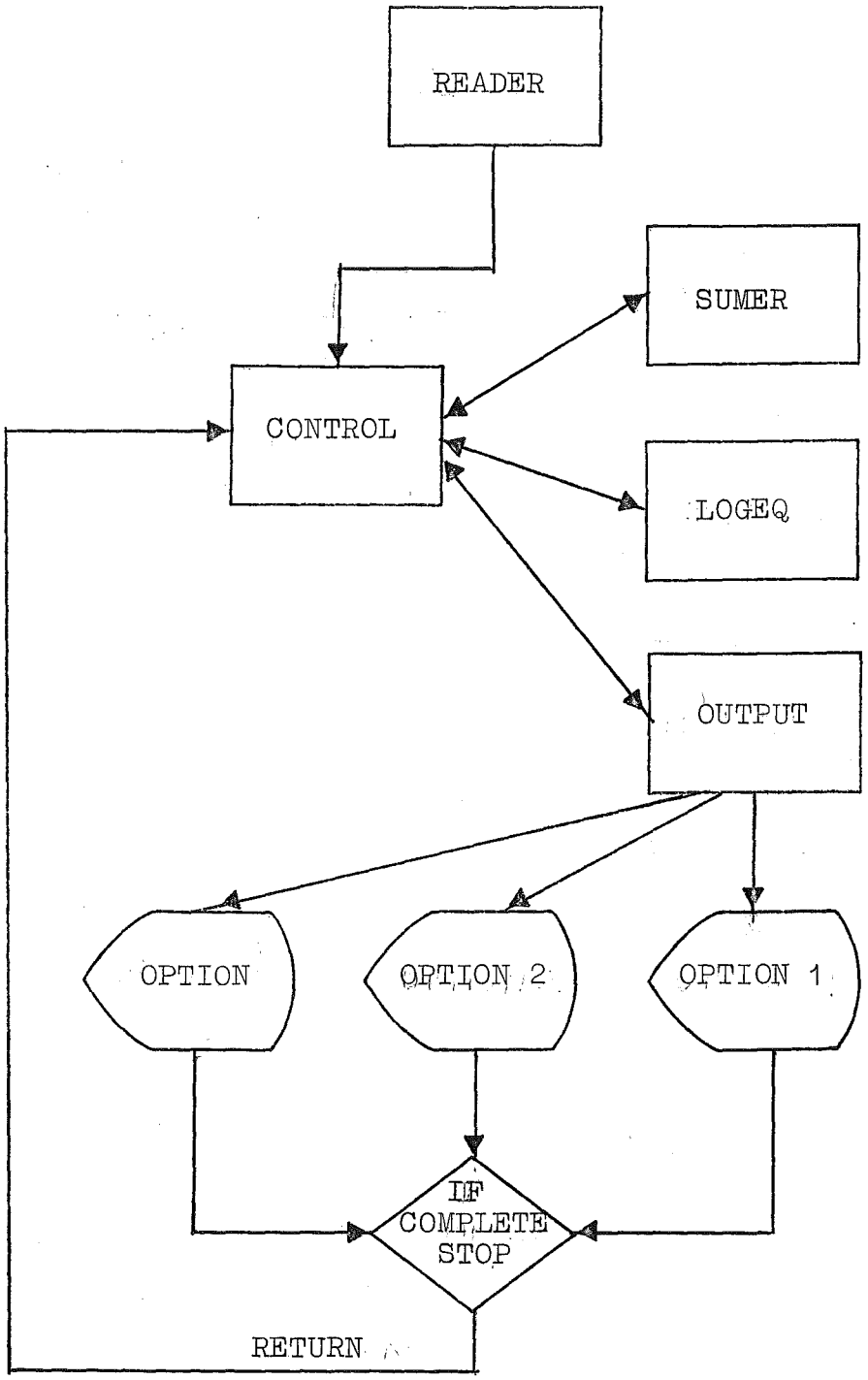


Figure 3.24

transfers computation to the correct one. Certain book-keeping is done in this program. In particular, failure counters and iteration numbers are updated. (60 Fortran Instructions)

SUMER:

This program determines if the output for each gate has been computed. If it has not, it checks to see if all of the input lines have been specified either as independent inputs or as outputs from other gates. When a gate can be computed, the gate number is transferred back to the control program which then selects the LOGEQ routine. (24 Fortran Instructions)

LOGEQ:

The logeq program computes the output gate value for the gate number passed from the sumer routine. The type of gate is written in literal form in the input viz., AND, OR, NOT and the appropriate function is selected. If the "present" computation is in the first stage, the variable OPT will select one of three options: the equation will be computed, it will be written and will be punched. The options are: 1., all three, 2., the first two, 3., the first one. A counter which says that the particular gate has been computed is then set. Control is transferred back to CONTROL. (70 Fortran Instructions)

OUTPUT:

If all gates have been computed, the total network has been computed. This condition causes control to be transferred to the output routine. There are three options. All three are variations of the fault table. Examples of the options will be given shortly. (190 Fortran Instructions)

3.7.3 Input

It was necessary to make certain assumptions about the number of gate inputs in order to simplify the data input. Because gates can have from two to six inputs in most of the hardware configurations, it was decided that three input gates would be an acceptable compromise. A two input gate could then be modelled by tying together two inputs and a six input gate could be modelled by two or more of the three input gates. Because of the program design, any of the common types of gates can be modelled and the program permits the user to insert models of gate types which are not present in the program. He need simply write the gate logic equation and convert it to a faultable gate model. It must be given a name e.g. FEQV (faultable equivalence gate); reference, though input or output, is always in literal format.

Those gates which have no connections within the given network block are termed independent inputs and are referred to by IDIX(I,J) where I is the particular gate terminal and J is the gate number.

The interconnection between gate outputs and other gate inputs is obtained through a topological connection matrix which is called CNX(I,J,K). If the output of gate I is connected to the Kth gate on its Jth terminal, a '1' is entered in the element (I,J,K). Otherwise a '0' is entered.

The proposition associated with the value (0 or 1) of each input is entered as a truth value T or as an F if the input is false. This is specified initially but may be changed either through automatic changes specified by the program or through a data card which can be read at the conclusion of a network computation.

Hence, it is possible to obtain the gate response to any set of inputs desired.

3.7.4 Output

Output is of two types. The first type is generated in the LOGEQ routine and is in the form of line printer output showing the expression for the logic equation and its output value or, in addition, it may be in punched card form. The second type of output is generated by the OUTPUT routine. The options are:

1. Input (Decimal) (Fixed) — Fault Condition — Output (Decimal)
2. Input (Decimal) (Variable) — Output (Decimal) — Fault Condition
3. Input (Decimal) (Variable) — Fault Condition — Fault Table (All Outputs)

The type of output desired can be selected by specifying the value of the variable OUT as 1,2, or 3. Illustrations of the output will be given later.

3.7.5 Program Operation

A brief account of the algorithm employed by the program is as follows:

1. Input data on gates, inputs, outputs desired, connection matrices and control variables are read.
2. The CONTROL program takes over when all data are read and establishes all initial array values.

3. The SUMER routine is called and a gate with a set of three independent inputs is located. This gate is 'marked'.

4. The LOGEQ routine is called by CONTROL and the marked gate is computed. The output from the LOGEQ program is that mentioned under Section 3.2.2 and is displayed the first time the gate is computed.

5. CONTROL establishes whether the all gates have been computed. If some gates remain uncomputed, the program goes back to step 3.

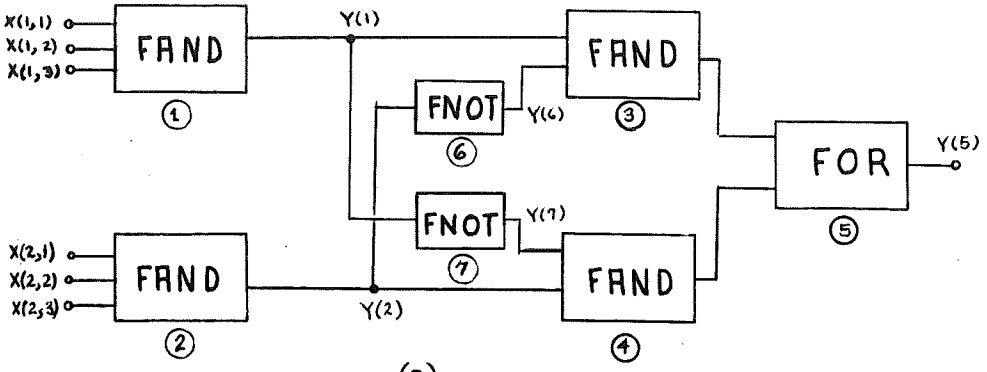
6. If all gates have been computed, CONTROL calls the output routine. Here the option (1,2, or 3) causes the line printer to output the information causes in the format desired.

7. CONTROL is again accessed and the faulting routine in the control program establishes a fault on a gate and the output for this condition is computed. The program cycles until all fault conditions and the resulting outputs have been computed.

This brief description of the program illustrates the basic computational procedure used. We now present the result for a particular network. It is essentially the network given in Figure 1.18.

3.8 EXAMPLE SIMULATION

The network for the example simulation is shown in Figure 3.25. It is the network in Figure 3.18 except with the INHIBIT gates replaced by FNOT and FAND gates. In addition, gates 1 and 2 are three-input gates. The individual gate expressions and their output value for the input vector shown at the top and all gates good is



(a)

INPUT VECTOR (1 1 1) OUTPUT VECTOR (1 1 1 1 1)

```
Y( 1) = B( 1).OR.A( 1).AND.X( 1,1).AND.X( 1,2).AND.X( 1,3)
Y( 1) = F
Y( 2) = B( 2).OR.A( 2).AND.X( 2,1).AND.X( 2,2).AND.X( 2,3)
Y( 2) = T
Y( 6) = B( 6).OR.(.NOT.X( 6,1).AND.A( 6))
Y( 6) = F
Y( 3) = B( 3).OR.A( 3).AND.X( 3,1).AND.X( 3,2).AND.X( 3,3)
Y( 3) = F
Y( 7) = B( 7).OR.(.NOT.X( 7,1).AND.A( 7))
Y( 7) = T
Y( 4) = B( 4).OR.A( 4).AND.X( 4,1).AND.X( 4,2).AND.X( 4,3)
Y( 4) = T
Y( 5) = B( 5).OR.(X( 5,1).OR.X( 5,2).OR.X( 5,3)).AND.A( 5)
Y( 5) = T
```

(b)

Figure 3.25

given in Figure 3.25(b). The equations are shown to have values T or F standing for 1 or 0. The output vector is $y_1y_2y_3y_4y_5y_6y_7$ = 0101101 for all gates good and for all independent inputs set to 0. For computer programming of the faultable equations, the letters A = γ and B = β are used. The symbols ., +, -, are replaced by their FORTRAN equivalents: .AND., .OR., .NOT. .

Note in addition that the subscripting on the inputs is (gate number, terminal number).

An example of each of the three output options listed in Section 3.7.4 is shown in Figure 3.26. The input vector in each case is 001111. The output vector for all gates good is 0101101. The first two options do not appear to be too much different. However, in option 2, the input will be automatically varied starting with the value 111111 and continuing 011111, 101111, etc. Output option 1 is a "once through" option with a particular set of inputs or usually, a single input. In each case, the decimal equivalent of the input and output vector is listed. The output for each fault condition is coded using a bit weighting 2,4,8,16 etc. "in that order". Coding for the input vector is ....,16,8,4,2,1 in that order with the right hand bit the least significant.

Option 3 is the really useful one for developing test type information. It shows the binary and decimal equivalent of the input vector and develops the fault table for a fixed input and each fault. The fault table is the modulo 2 sum of the good output vector and the vector resulting from the given input but containing the fault specified to the left of the table. Each output appears in the table and the ordering is the same as for the output vector. For example, in Figure 3.26,

```
INPUT VECTOR        0C1111                                  OUTPUT VECTOR      01011 1

        X(I)-BASE TEN       Y(I)-BASE TEN       FAILURE TYPE
                 15                 18C        NETWORK FAULT-FREE
                 15                 18C          GATE( 1)  S-A-0
                 15                 192          GATE( 2)  S-A-0
                 15                 18C          GATE( 3)  S-A-0
                 15                 132          GATE( 4)  S-A-0
                 15                 14P          GATE( 5)  S-A-0
                 15                 18           GATE( 6)  S-A-0
                 15                  4           GATE( 7)  S-A-0
                 15                  6           GATE( 1)  S-A-1
                 15                 18C          GATE( 2)  S-A-1
                 15                 188          GATE( 3)  S-A-1
                 15                 18C          GATE( 4)  S-A-1
                 15                 18           GATE( 5)  S-A-1
                 15                 244          GATE( 6)  S-A-1
                 15                 18           GATE( 7)  S-A-1

END OF PROGRAM

                        OUTPUT OPTION IS ---  OUT = 1

INPUT VECTOR        0C1111                                  OUTPUT VECTOR      01011C1

                INPUT - OUTPUT VECTORS FOR THE GOOD MACHINE
                 15                 18C        NETWORK FAULT-FREE
                INPUT - OUTPUT VECTORS FOR FAILED MACHINES
                 15                 18C          GATE( 1)  S-A-0
                 15                 192          GATE( 2)  S-A-0
                 15                 18C          GATE( 3)  S-A-0
                 15                 132          GATE( 4)  S-A-0
                 15                 148          GATE( 5)  S-A-0
                 15                 18           GATE( 6)  S-A-0
                 15                  4           GATE( 7)  S-A-0
                 15                  6           GATE( 1)  S-A-1
                 15                 18C          GATE( 2)  S-A-1
                 15                 188          GATE( 3)  S-A-1
                 15                 18C          GATE( 4)  S-A-1
                 15                 18           GATE( 5)  S-A-1
                 15                 244          GATE( 6)  S-A-1
                 15                 18           GATE( 7)  S-A-1
END OF PROGRAM

                        OUTPUT OPTION IS ---  OUT = 2

INPUT VECTOR        0C1111                                  OUTPUT VECTOR      01011 1

      INPUT              FAULT                                  UNLIKE OUTPUTS  SUM

                 15                ALL GATES-GOOD                 0000000        7
                 15   GATE  1 STUCK-AT-ZERO                       0000000        7
                 15   GATE  2 STUCK-AT-ZERO                       0101111        3
                 15   GATE  3 STUCK-AT-ZERO                       0000000        7
                 15   GATE  4 STUCK-AT-ZERO                       0001101        5
                 15   GATE  5 STUCK-AT-ZERO                       0000101        6
                 15   GATE  6 STUCK-AT-ZERO                       0000000        7
                 15   GATE  7 STUCK-AT-ZERO                       0001101        4
                 15   GATE  1   STUCK-AT-ONE                      1001101        3
                 15   GATE  2   STUCK-AT-ONE                      0000000        7
                 15   GATE  3   STUCK-AT-ONE                      0010000        6
                 15   GATE  4   STUCK-AT-ONE                      0000000        7
                 15   GATE  5   STUCK-AT-ONE                      0000000        7
                 15   GATE  6   STUCK-AT-ONE                      0001101        6
                 15   GATE  7   STUCK-AT-ONE                      0000000        7

                END OF PROGRAM
END OF PROGRAM

                        OUTPUT OPTION IS ---  OUT = 3
```

LOGEQ Display Options

Figure 3.26

the output with all gates good gives outputs which are no different from the output vector 0101101. However, with gate 2 stuck-at-zero, the output is 0000011. Hence the fault table entry is computed by the modulo 2 sum given by

$$\begin{array}{rl} & 0101101 \\ \oplus & 0000011 \\ \hline = & 0101110 \end{array}$$

which can be verified as correct from the figure. A figure of merit labelled SUM is listed on the side. It is the total number of outputs minus number of outputs which are different from the good output for the given fault. It may be used as a measure of the network's sensitivity to a particular fault for a given input.

Application of Fault Table Option

The information in the fault table option may be used for test type specification. We will now describe this application. For each input vector, the columns of the fault table list the value of the outputs for all faults. Each time the particular output is different from the good output, a 1 appears in the fault table. For a given input, the best output to select for test purposes is the one which has the greatest number of ones.

In the fault table option OUT = 3 in Figure 3.26, two columns - number 4 and 5 - corresponding to outputs 4 and 5 contain four and five ones respectively. This means that for the input 001111 which has a decimal equivalent of 15, only four of the fourteen possible faults will be detected by monitoring gate 4 output or five by monitoring

gate 5 output.

In Figure 3.25(a) the network is essentially equivalent to the network shown in Figure 3.18. Consequently, the input used to test the two networks should be identical. That this is so can be verified from Figure 3.27. The fault tables for the inputs 110111, 111011 and 111111 are shown. These correspond to the 0x11, 110x and 1111 inputs illustrated in Figure 3.23 for the four input gates. x means "don't care". Scanning row five of the fault table in Figure 3.27, for an input of 110111, the faults detected are (2,s-a-0), (4,s-a-0), (5,s-a-0) and (1,s-a-1). (We can disregard gates 6 and 7 for this discussion because they simply form a part of the inhibit gate in Figure 3.18). In figure 3.23, we notice that input 0x11 also detects faults (2,s-a-0), (4,s-a-0), (5,s-a-0) and (1,s-a-1). Similarly, the input 110x and 111011 can be shown to give equivalent fault detection indications as do 1111 and 111111.

In both of these networks, (Figure 3.18 and Figure 3.25) gate number 5 output is the most useful for obtaining fault detection information. Although this example is simple, it illustrates the application of the simulation and verification of its operation can be obtained. Its real application is to longer multiple output networks. Two networks of twenty or more gates were simulated using this program. The limitation seems to be not in generating the fault table but in displaying the results. The two options shown in Figure 3.26 give a decimal output which is more compact but not so useful for test input specification as the fault table option.

INPUT VECTOR 110111 OUTPUT VECTOR 0101101

INPUT	FAULT			UNLIKE OUTPUTS	SUM
[illegible]	55		ALL GATES-GOOD	[illegible]	7
[illegible]	55	GATE	1 STUCK-AT-ZERO	[illegible]	7
[illegible]	55	GATE	2 STUCK-AT-ZERO	0101110	3
[illegible]	55	GATE	3 STUCK-AT-ZERO	[illegible]	7
[illegible]	55	GATE	4 STUCK-AT-ZERO	[illegible]	5
[illegible]	55	GATE	5 STUCK-AT-ZERO	[illegible]	6
[illegible]	55	GATE	6 STUCK-AT-ZERO	[illegible]	7
[illegible]	55	GATE	7 STUCK-AT-ZERO	[illegible]	4
[illegible]	55	GATE	1 STUCK-AT-ONE	1001101	3
[illegible]	55	GATE	2 STUCK-AT-ONE	[illegible]	7
[illegible]	55	GATE	3 STUCK-AT-ONE	[illegible]	6
[illegible]	55	GATE	4 STUCK-AT-ONE	[illegible]	7
[illegible]	55	GATE	5 STUCK-AT-ONE	[illegible]	7
[illegible]	55	GATE	6 STUCK-AT-ONE	[illegible]	6
[illegible]	55	GATE	7 STUCK-AT-ONE	[illegible]	7

INPUT VECTOR 111011 OUTPUT VECTOR 1010110

INPUT	FAULT			UNLIKE OUTPUTS	SUM
[illegible]	59		ALL GATES-GOOD	[illegible]	7
[illegible]	59	GATE	1 STUCK-AT-ZERO	1010101	3
[illegible]	59	GATE	2 STUCK-AT-ZERO	[illegible]	7
[illegible]	59	GATE	3 STUCK-AT-ZERO	[illegible]	5
[illegible]	59	GATE	4 STUCK-AT-ZERO	[illegible]	7
[illegible]	59	GATE	5 STUCK-AT-ZERO	[illegible]	6
[illegible]	59	GATE	6 STUCK-AT-ZERO	[illegible]	4
[illegible]	59	GATE	7 STUCK-AT-ZERO	[illegible]	7
[illegible]	59	GATE	1 STUCK-AT-ONE	[illegible]	7
[illegible]	59	GATE	2 STUCK-AT-ONE	[illegible]	3
[illegible]	59	GATE	3 STUCK-AT-ONE	[illegible]	7
[illegible]	59	GATE	4 STUCK-AT-ONE	[illegible]	6
[illegible]	59	GATE	5 STUCK-AT-ONE	[illegible]	7
[illegible]	59	GATE	6 STUCK-AT-ONE	[illegible]	7
[illegible]	59	GATE	7 STUCK-AT-ONE	[illegible]	6

INPUT VECTOR 111111 OUTPUT VECTOR 1100000

INPUT	FAULT			UNLIKE OUTPUTS	SUM
[illegible]	63		ALL GATES-GOOD	[illegible]	7
[illegible]	63	GATE	1 STUCK-AT-ZERO	1001101	3
[illegible]	63	GATE	2 STUCK-AT-ZERO	0110110	3
[illegible]	63	GATE	3 STUCK-AT-ZERO	[illegible]	7
[illegible]	63	GATE	4 STUCK-AT-ZERO	[illegible]	7
[illegible]	63	GATE	5 STUCK-AT-ZERO	[illegible]	7
[illegible]	63	GATE	6 STUCK-AT-ZERO	[illegible]	7
[illegible]	63	GATE	7 STUCK-AT-ZERO	[illegible]	7
[illegible]	63	GATE	1 STUCK-AT-ONE	[illegible]	7
[illegible]	63	GATE	2 STUCK-AT-ONE	[illegible]	7
[illegible]	63	GATE	3 STUCK-AT-ONE	[illegible]	5
[illegible]	63	GATE	4 STUCK-AT-ONE	[illegible]	5
[illegible]	63	GATE	5 STUCK-AT-ONE	[illegible]	6
[illegible]	63	GATE	6 STUCK-AT-ONE	[illegible]	4
[illegible]	63	GATE	7 STUCK-AT-ONE	[illegible]	4

END OF PROGRAM

Fault Table Printout OUT=3

Figure 3.27

3.9 SUMMARY AND CONCLUSIONS

A discussion of the procedures for developing diagnostic information for combinational networks has been presented. The paramount objective of any fault detection or isolation technique is to develop the shortest possible test. This means that the minimum number of input combinations which will detect and/or isolate all faults is required.

It was shown that the easiest way to perform fault isolation is to use serial testing techniques based on a sequence of tests. Here, efficiency is important and the effectiveness of a particular sequence must be evaluated to determine the degree of optimality.

Three contributions have been made. These are a) a statement of an efficiency criterion for multiple output tests, b) an analysis procedure for determining optimal fault detection tests and c) a network simulator designed for fault studies.

The fault analysis procedure yields an expression for the output function in terms of input signal propositions and element fault propositions. The network output may be found by fixing the values of the fault parameters γ and β. Using the output proposition and its complement, all single and multiple faults of the class stipulated may be simulated and tests for these developed. We have mainly considered single faults. The appearance of network path terms in the sum-of-products expression was mentioned as being useful for tracing signal paths. An example which illustrates the method was presented.

The simulation which was developed to study the

effect of faults on the outputs in the network was presented. It can be used to obtain test input information for single or multiple output networks. An example which showed that it gives results equivalent to the analytical procedure was presented.

Using the simulation and the various options which were described, fault data can be developed for a fault dictionary which can be used in an automatic checkout system. Alternatively, the simulation can be used to produce information for developing optimal test input combinations for fault detection. Using the fault detection test information, serial tests for fault isolation can be derived by applying a testing diagram.

CHAPTER 4 PAGE NO.

4 SEQUENTIAL NETWORKS

4.0 INTRODUCTION

Fault detection and isolation in finite state, binary sequential networks has been described by various authors[34,39]. These descriptions range from the abstract sequential machine experiments[107,21] to diagnosis of total sequential switching systems[37,52,88,128]. In this chapter, the specification of sequences of discrete value input signal combinations useful for fault detection and perhaps isolation in sequential networks is discussed. For any given input sequence, the output sequence for the good network is compared with the output from a questionable network. If any element in the output sequences is different, the fault will be detected. We show that it is possible to extend this comparison to develop fault isolation for some faults.

The effectiveness of a given input sequence for test purposes can be evaulated by computing the number of faults detected (isolated) out of the total possible. An empirical figure of merit μ for a given input sequence length SL, is

$$\mu = \frac{TF \cdot FD}{SL(TF - FD + 1)}, \qquad SL \geq 1$$

where TF is the total faults and FD is the faults detected. The larger the value of μ, the more optimal the input sequence for detecting the given class of faults. This figure of merit can be used to compare the effectiveness of any two or more sequences when applied to a given network. If desired, FD can be replaced by fault isolation, FI.

Some authors have failed to establish an important distinction between sequential machines and sequential networks or circuits. Understanding the difference is important in discussions on diagnosis. A sequential machine is an abstract representation of a sequential network. The possibility of describing a machine which has no physical counterpart is not to be discounted. On the other hand, the network is an electrical analogue or realization of the machine. Its external behaviour is equivalent to that specified by the machine. This means that if a given sequence of input patterns is applied to the machine and also to its network realization, the output sequences will be identical if the machine and network have been properly designed and are fault free. The machine signal representation is in terms of symbols whereas the network signals are voltage or current levels.

Many sequential networks are synchronous where synchronization is provided by an external clock. In this chapter, we will treat those networks constructed from gates and clocked memory elements. It is stipulated that the combinational logic be composed of AND/OR/NOT logical elements. Memory elements of the RESET/TRIGGER type will be used. Although practical networks use NAND/NOR logic and various memory elements such as J-K and set dominant flipflops, we can specify the above types as representative of typical sequential network components without loss of generality.

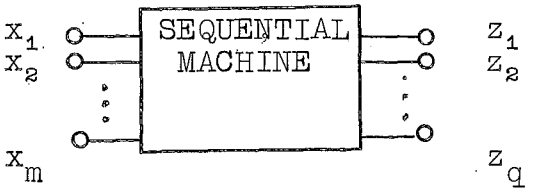
Implicit in all of the discussion is the limiting of faults to those which occur in the logical functioning of the network gates or in the function of the memory elements. Also, all machines and networks are finite state[56].

Two ideas are introduced in this chapter which are potentially useful for developing efficient sequential network test sequences. The first is the state exercising sequence. It is a sequence of input combinations designed to cause changes in memory element states and which is potentially shorter than the conventional distinguishing sequence[23]. The distinguishing sequence does not take into account any physical characteristics of the network which may lead to a requirement for shorter test sequences. This is to be expected because distinguishing sequences are defined for sequential machines. The second idea is that of state variable observability. A machine which is completely state observable has an output function which depends on each state variable. It is potentially easier to isolate some faults if the output is an explicit function of the inputs and states. A machine whose output depends on both inputs and states is a Mealy machine[92]; Moore machines[107] have an output dependent on the states only. Most practical networks can be abstractly described by either machine.

In Section 4.1, models of synchronous sequential networks are introduced. This section is followed by a discussion on the sequential network functional element diagnostic models in Section 4.2. The distinction between sequential networks and sequential machines is made in Section 4.3 and the preliminaries for Section 4.6 are discussed in Sections 4.4 and 4.5 which are on state exercising sequences and state variable observability respectively. Following the test development discussion in Section 4.6, a short discussion on empirical-experimental test development methods is given in Section 4.7 which concludes the chapter.

4.1 MODELS OF SYNCHRONOUS SEQUENTIAL NETWORKS

An external description of a sequential machine is shown in Figure 4.0. The input signal levels $X = \{x_1, x_2, \ldots, x_m\}$ and output signal levels $Z = \{z_1, z_2, \ldots, z_q\}$ are vectors whose elements come from the set $B = \{0,1\}$. X is applied to m input terminals and



External Description of a Sequential Machine

Figure 4.0

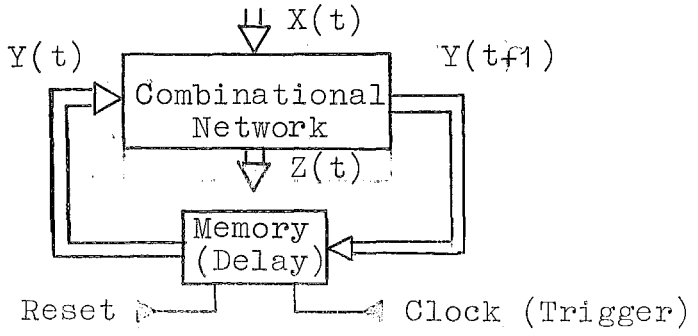
Z is observed or computed at the q output terminals. Each possible combination of signal levels will be called a signal value. For simplicity, we will use input and output to mean input signal value and output signal value respectively.

The symbols X(t) and Z(t) will denote the input and output at time t. Because the networks which we will be treating are synchronous, it is convenient to restrict observation times to the instants specified by $t = 1,2,3,\ldots$. It is assumed that these times correspond to clock pulses.

The fact that clock signals are pulses means that although X and Z are restricted to 0 or 1 levels, the clock and possibly reset are pulse or return-to-zero signals. Pulse signals will be used exclusively for the

timing and reset functions and need not be further discussed.

In Figure 4.1, an internal description of the sequential machine is given which affords a finer description than the external description. A yet to be



Internal Description of Sequential Machine

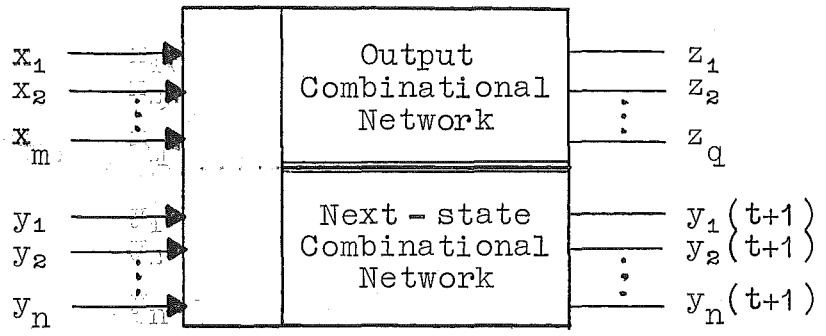
Figure 4.1

defined set of signals appears on the input and output terminals of the memory unit. These are the sequential network <u>state signals</u> and are written $Y = \{y_1, y_2, \ldots y_n\}$. The state at any time t is written Y(t). The relationship between the input, state, and output signals can be written as mappings given by

$$\begin{aligned} \omega &: X \times Y \rightarrow Z \\ \sigma &: X \times Y \rightarrow Y(t+1) \end{aligned} \qquad 4.1$$

where ω is the <u>output function</u> or mapping and σ is the <u>next state mapping</u>. Y(t+1) (or Y') is the <u>next state</u>.

ω and σ can be equated with the output assignment and state assignment Boolean functions. This equivalence is shown in Figure 4.2(a). The combinational network portion of Figure 4.2 is divided into the output combinational network and the next-state combinational network. The equations show that ω is given explicitly in terms of F_{B_Z} and σ is given by F_{B_Y}. The delay unit



$$\text{Inputs} = X = (x_1, x_2, \ldots, x_j, \ldots, x_m)$$

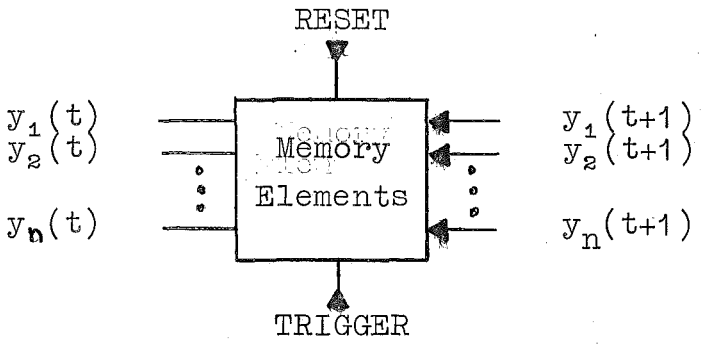
$$\text{States} = Y = (y_1, y_2, \ldots, y_k, \ldots, y_n)$$

$$\text{Outputs} = Z_1 = (z_1, z_2, \ldots, z_l, \ldots, z_q)$$

$$Z = F_{B_Z}(X, Y)$$

$$Y(t+1) = F_{B_Y}(X, Y) \qquad \text{(a)}$$

Combinational Circuitry



Memory Network (b)

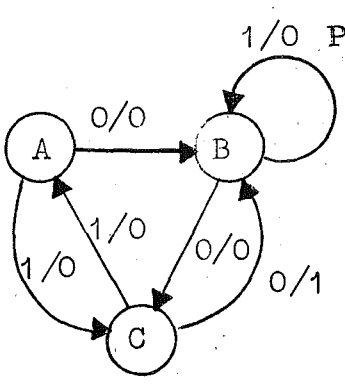
Figure 4.2

in Figure 4.1 is shown in more detail in Figure 4.2 (b). It is assumed that each state variable y_i^k is delayed one unit of clock time by one of the memory elements.

The cause-effect or input-output relationships among input, state, next state and output given in Equation 4.1 are described by a state diagram or state table[56]. An example of a state transition diagram for a sequential machine is shown in Figure 4.3(a). The diagram is for a network with three states A,B and C. The arrow direction indicates a state transition. For example a state which changes from A to B under the input signal 0 and with an output 0 is shown at the top of the diagram. There is no direct transition from B to A. In this type of diagram the branch labelling is "Input/Output". The state transition table shown in Figure 4.3(b) is equivalent to the diagram in Figure 4.3(a). If a network has k terminals, the state table will contain 2^k columns. The network representation for the machine described by Figure 4.3 will have a single input terminal, a single output terminal and at least two state variables.

The network output sequence depends on the starting state[56] and the input sequence. This can be shown using the example machine in Figure 4.3. Suppose that the machine represented by Figure 4.3 has the input sequence 001011 applied to its input terminal. If the starting state is A, the output is 000000 and the state sequence is ABCABBB. Similarly for starting state B, the output is 010000 and for C, 100000. These can be verified using either the diagram of table.

An essential difference between the machine and its network representation is the state assignment[108,77]. In the machine description in Figure 4.3, the general states



State Transisition Diagram

(a)

Present State	Input 0	Input 1
A	B 0	C 0
B	C 0	B 0
C	B 1	A 0

State Transition Table

(b)

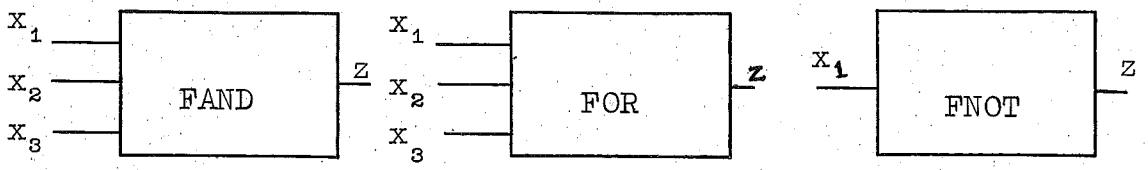
Figure 4.3

are A,B and C. A state assignment specifies the general state as a combination of network state variable values. If the number of state variables is n, there are 2^n different state combinations possible. This will require n memory elements. The number 2^n must be equal to or greater than the number of general machine states. A possible assignment for the states in Figure 4.3 would be A = 11, B = 00, C = 01. A primary sequential network design problem is deciding how to allocate these network states.

4.2 DIAGNOSTIC MODELS FOR SEQUENTIAL NETWORK ELEMENTS

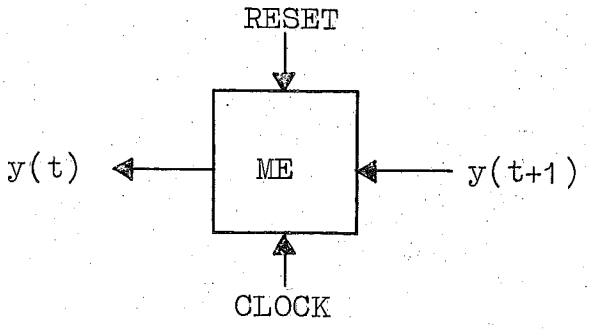
The FE diagnostic models from which all synchronous sequential networks may be constructed are shown in Figure 4.4. Combinational elements are the FAND and FOR type presented in Figure 3.17 in Chapter 3 and the FNOT gate is the faultable NOT gate. The memory element shown in Figure 4.4 is representative of a class of units used in network designs. Although each type of memory element has individual differences and properties which may be desirable for particular applications, the element in Figure 4.4 possesses the memory, delay, clocking and reset properties of more complex elements. It may be observed from the function table in Figure 4.4 that the element is basically a delay element with reset override. That is, if the reset is activated, the next output will be low no matter what the present input may be.

Although logical variables can be appended to the memory element to simulate logical faults, we will not incorporate this refinement here. However, if the memory element is to be applied in a simulation, this alteration to the function can be made. We have shown stuck-at-zero



$z = x_1 x_2 x_3 \gamma + \beta$ $z = (x_1 + x_2 + x_3)\gamma + \beta$ $z = \bar{x}_1 \gamma + \beta$

Combinational Elements



y_c	r_c	c	y	y^0	y^1
0	0	1	0	0	1
0	1	1	0	0	1
1	0	1	1	0	1
1	1	1	0	0	1
Present Value		Next Value			

KEY:

c	CLOCK
r	RESET
ME	Memory Element
1	High (TRUE)
0	Low (FALSE)

The output appears a short time after c is applied.

Memory Element

Figure 4.4

and stuck-at-one faults as y^0 and y^1 in the table in Figure 4.4.

4.2.0 Combinational Element Faults

The types of combinational element faults postulated are:

1. Output s-a-1
2. Output s-a-0

The properties of the gates and the methods for simulating these faults were covered in Chapter 3. Justification for assuming this class of faults was also given there.

4.2.1 Memory Element Faults

All memory units are assumed to be reset-trigger units and are assumed to develop faults of the following types.

1. The element will not reset. It stays s-a-0 or s-a-1 on the reset pulse. It may still trigger properly. If it triggers correctly then, the s-a-0 fault is an "artificial" fault and not detectable.
2. The element will trigger from the reset state but stays s-a-1 until a reset pulse is applied.

These faults are summarised in the table in Figure 4.5. The values for y(t) are given in each column.

The faults s-a-0 and s-a-1 are the easiest to detect and isolate. The T.F.-1 (trigger fault s-a-1) is more difficult to detect because it will not appear until the second trigger is applied. This might be well along in

Table of Faults and Effect on y(t)

RESET or Output	NOTRIGGER	TRIGGER (t=1)	TRIGGER (t=2)	FAULT TYPE
0	0	0	0	s-a-0
1	1	1	1	s-a-1
0	0	1	1	T.F.-1
1 0	1 0	0 1	1 0	INTERMITTENT Reset Fault

Figure 4.5

the input sequence. The third type is the intermittent which fails to respond to the reset pulse. It is termed intermittent because it may or may not affect the output, depending on whether the output was 0 or 1 on the application of the input reset pulse.

4.3 MACHINE EXPERIMENTS VERSUS NETWORK DIAGNOSIS

The properties of a distinguishing sequence have been described by Moore[107] and applied by Hennie[56] and Kime[71] to sequential machine experiments. Kohavi Lavallee[72] and Kohavi and Kohavi[73] have extended the application to sequential machine design with fault detection capabilities and have introduced the variable length distinguishing sequence.

A sequence of input combinations which will produce a different output sequence for each initial state is called a distinguishing sequence. By applying a distinguishing sequence to a sequential machine and observing the resulting output, it is possible to determine the initial state of the machine.

In developing checking experiments for sequential machines, Hennie[56] has assumed that an experiment can be designed to determine that the machine is good if the following conditions apply: (a) the machine has a distinguishing sequence, (b) no two states are equivalent (the machines is reduced), and (c) the machine is strongly connected. If these conditions apply, then a checking experiment will consist of an input sequence which causes the machine to perform a transition from each state to every other state. The experiment usually begins by applying a homing sequence which puts the machine in a known state. Starting from the known initial state, a distinguishing sequence is applied and the corresponding outputs are observed. Any state transitions not covered by the distinguishing sequence are then verified by applying an appropriate sequence. If the outputs for the given inputs are correct, which can be determined from the state diagram, then the machine is judged good. Any deviation from the good outputs during the course of the experiment causes the machine to be judged bad. At any stage during the experiment if all outputs have been the correct ones, the machine is judged possibly good.

Sequential network diagnostic test procedures have been developed from the machine checking experiments. In a checking experiment, no knowledge of the actual circuitry in the network is assumed. Indeed, a checking experiment as defined by Hennie is independent of any particular circuit. For this reason, it is presumptuous to assume that machine diagnostic procedures applied to networks will be efficient. Needless to say, a machine experiment cannot be adopted to fault isolation. A synchronous network feature which is potentially useful for decreasing

the test length is the capability of selecting the initial state *. This freedom eliminates the necessity for applying a homing sequence. Another questionable requirement of the machine experiment is the necessity for verifying each state transition. From physical considerations, one can argue that if each memory element transition can be verified, the memory section of the network is good. For example, it may be sufficient to apply a sequence which verifies that each memory element is working properly. If all elements reset to a given state and if they perform the required transitions, they are judged good. Because more than one state variable may change at a time in a synchronous network, sequences which cause the greatest number of state variable transitions are the most desirable. Asynchronous networks are designed to have only single state variable transitions in order to eliminate races or hazards [34,60,93]. Consequently, shortening test lengths by exciting several transitions is not possible for these networks.

Using a circuit oriented approach, a synchronous network test should consist of applying input sequences which will verify proper operation by checking

1. Output logic
2. Next state logic
3. Transition of memory units and the RESET state.

The main problem in synchronous sequential network diagnosis is to specify input sequences which will verify

---

* In asynchronous network diagnosis, the ability to reset the memory units to a known initial state may not be possible. For this reason, it is necessary to apply a homing sequence as the first input sequence in asynchronous network testing to put the network in a known state.

correct operation of each of the two logic networks and the memory units and which are short enough for practical application[71]. There are two distinct but interrelated facets to this problem. One is the limitation on independently selecting the inputs to the output logic. The other is the fact that only indirect observation of the next state logic outputs can be made.

We propose in this chapter to treat these two facets by devising input sequences which will verify each section independently assuming that the other two sections are fault free. If desired, an intersection process can be applied which detects and eliminates redundant input sequences and hence reduces the total number of terms in the testing sequence. This can lead to a potentially short test sequence.

In the next two sections we introduce two useful heuristics. The first is the state exercising sequence. This is an input sequence which will cause the memory elements to cycle through their possible transitions. In Section 4.5, the idea of state variable observability is introduced.

4.4 STATE EXERCISING SEQUENCES

A state exercising sequence is an input sequence which causes the state variables to make the transitions, $0 \rightarrow 1 \rightarrow 0$ or $1 \rightarrow 0 \rightarrow 1$ at least once. If in a synchronous network, the reset state is assumed to be the starting state, the variables will be required to go from 0 to 1 and back to 0 (assuming all resets are 0). The input sequence (X) which causes the variables in the set $\{y_1 y_2 y_3 \ldots y_n\}$ to change through the values $0 \rightarrow 1 \rightarrow 0$

is called a state exercising sequence.

Heuristically, if the output is correct for each state in the sequence, then the memory units are probably good. This follows from the fact that if all memory units have gone through one complete cycle, then the only sources of errors will be in other state variable transitions. These errors would be those caused by errors in the next state logic.

Consider the Moore machine-developed-network. The output is a function of the states only. This means that if for a given state exercising sequence, the states are set sequentially and one-at-a-time faults are tested by using a Boolean difference[1125] then the set of memory units which will be verified in each step can be written as $\{y\}_v$. If there are N combinations in a sequence, then the intersection of the N $\{y\}_v$ sets can be taken. If there is an element in the set of state variables Y not in this intersection then the particular state variable exercising sequence will not test for this fault. This possibility amounts to the particular variable behaving like a don't care conditions or a redundant variable (in the prime implicant sense) for this particular state variable exercising sequence. Basically, this means that errors in the memory are not necessarily detectable for the particular state exercising sequence.

For the Mealy designed network, the output is a function of both the state and the input. In this case, detection of faults both in the states and the input values is possible. If the inputs are assumed to be error free, then the same procedure used for error detection in the Moore network may be applied. To design a state exercising sequence, a state table which

shows the state variable values for each state is useful.

4.5 NETWORK STATE OBSERVABILITY

A network which contains all y_j in the output function equation is said to be completely observable. The implications of this condition can be clarified by the following discussion.

Suppose that a given input vector X_1 is applied to the combinational network and Y_1 is the present state. If the network is functioning properly, the next state and output will be predictable. For a subclass of errors in the state vector, the output will not change from its good value. This means that more than one combination of the y_j in the state vector will give the same output when the particular input pattern is X_1. However, certain of the incorrect states will cause the output to be in error. The particular set of state errors that changes the output under the particular input combination will be termed the output observable state vector errors under the input X_1. If a sequence of inputs exists for which all possible state transition errors are detectable, the network is said to be observable for the class of errors and the errors are detectable for the particular input patterns.

A Moore network is completely state variable observable. This means that the output is a function of each state variable. Unless the network is redundant, the value of each variable affects the value of the output. If the output function is realized by a canonical gate network[77], each term in the sum of products expansion is represented by a separate gate. In this case, each

term of the canonical expansion can be tested separately.

We will assume that the networks discussed have been designed and that a schematic diagram or the logical equations are available. If we were at liberty to specify how a network should be designed so that its properties were optimum for testing, then a minimal length state exercising sequence and complete state variable observability would be two important criteria. The possibility of "design for testing", particularly for the state assignment networks also merits further study.

4.6 TEST DEVELOPMENT

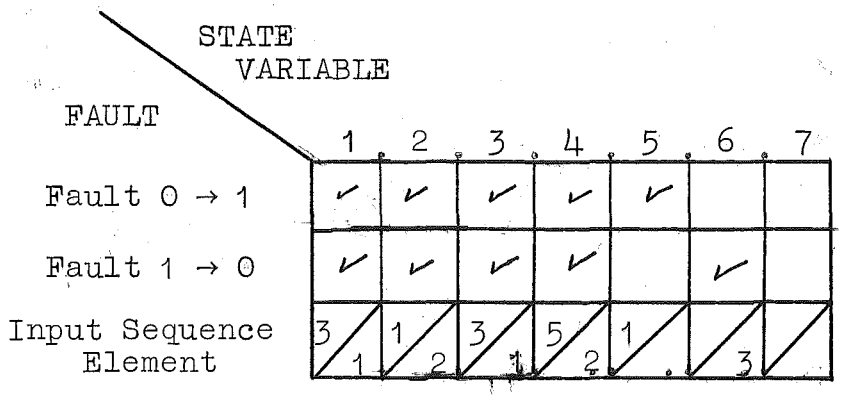
4.6.0 Treating Memory Faults

A sequential network will be denoted by M. The output of M for a particular input-state combination will be written M^{XY}. The reset state will be written as Y_o to indicate that all states are reset to zero. For a particular input reset combination $X_i Y_o$, the output will $M^{X_i Y_o}$. All one-at-a-time changes in the state variable values will be denoted by c(Y).

A state exercising sequence can be applied which starts with the reset state. For each step of the sequence, the output M^{XY} is computed. Then $M^{Xc(Y)}$ is computed. If any change c_i causes at least one of the outputs* to be different, this will be written $\overline{M}^{Xc_i(Y)}$. If changes are made sequentially and there are n state variables, a table of transition faults can

---

* The network may have more than one output terminal.



Memory Transition Fault Detection Table, Input = $(X)_{se}$

Figure 4.6

be compiled like the one shown in Figure 4.6. This table contains a tick if a transition change fault will be detected by an output discrepancy. The input sequence interval when the fault is detected is shown in the third row of the table in Figure 4.6. The entry above the diagonal applies to the 0 → 1 fault and the entry below the diagonal refers to the 1 → 0 fault. No entry indicates that the fault cannot be detected by using the particular sequence $(X)_{se}$. As an example, state variable No. 4 with a transition fault 0 → 1 is detected on the 5th input pattern.

4.6.1 Treating Logic Faults

The difficulty in detecting faults in the output logic has been mentioned as being that of generating the proper input-state signals. The required XY combinations can be specified using methods discussed in Chapter 3. The difficulty lies in selecting a sequence of X(t) such that Y(t+1) is the desired state.

It is difficult to detect faults in the next state logic because the output from this logic is delayed one unit of time by the memory element. Consequently, the following procedure is required:

1. Ascertain as far as possible that the present state is correct. This can be done using memory fault checks mentioned previously.
2. Using the present known state, select an input X which will test a portion of the next state logic. If the next state logic output is incorrect because of a logic fault, this fault will appear as an error in the next state variable corresponding to that output.
3. The next input applied is one which will give the correct output if the state is the good state and an incorrect output if the previous state logic variable mapping was erroneous.

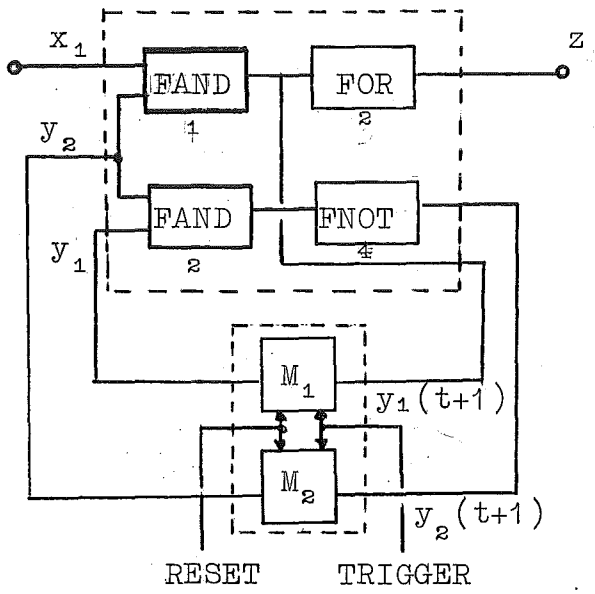
In the next section, we show how these ideas may be adapted to fault detection and isolation in a network which has been designed according to Gray code transitions.

4.6.2 Test Development Procedure

The combinational network shown in Figure 4.7(a) is state observable.* It has a state diagram shown in Figure 4.7(b). This can be verified by the state and output equations shown in Figure 4.7(d). The state table (Figure 4.7(d)) has been included for convenience. The state coding shown in Figure 4.7(c) is the Gray type code mentioned previously.

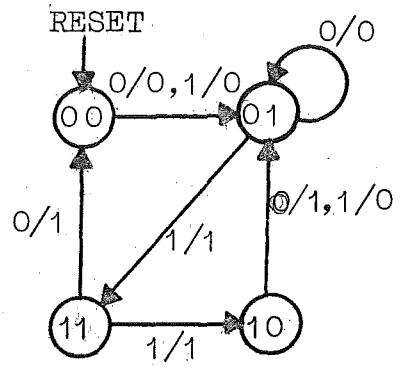
The test procedure follows the sequential test defined in Chapter 1. Here we use the state table to develop a

*A stronger observability occurs when y and y appear in the MSP $(^{77})$ F_{B_Z}.



Network Showing Logic and Memory

(a)



State Transition Diagram for Network shown in (a)

(b)

$$y_1(t+1) = xy_2$$

$$y_2(t+1) = \overline{y_1 y_2}$$

$$z = xy_2 + y_1 y_2$$

State and Output Equations

(c)

STATE \ INPUT	0	1
A	B 0	B 0
B	B 0	C 1
C	A 1	D 1
D	B 0	B 0

State Table

(d)

	y	y
A	0	0
B	0	1
C	1	1
D	1	0

State Assignment Coding

(e)

Figure 4.7

state exercising sequence which is a distinguishing sequence. The network is strongly connected and reduced. The state exercising input sequence which will be applied is 0110101.

If the network is reset, $y_1 = 0$ and $y_2 = 0$ and the above sequence is applied, the output sequence which results is RESET 0 : 0110110. The top row in the output assignment fault chart shown in Figure 4.8(a) lists the FAULT, INPUTS and REMARKS. The inputs to the network when it is good are shown in the row labelled GOOD under REMARKS. The xy_1y_2 vectors for the 7 interval sequence are 000,101,111,010,101,011,100. The outputs for the fault free network are shown immediately below. R.S. means reset. The outputs obtained for the output assignment network containing various types of faults are shown in the respective rows. The REMARKS column shows the time interval output value corresponding to the particular fault. For example, gate number 2 stuck-at-zero will cause the output during the 6th time interval to be 0 rather than 1. The stuck-at-one faults all have the same effect on the output. It stays at 1 no matter what the input. Note that the outputs are terminated at the point at which a fault is detected. This makes it possible to use the information for fault isolation simply by recording the time interval at which the output differs from the good output. In fact, it may be possible to use all of the output sequence to obtain information which is unique to the various faults by using a test diagram. This has not been illustrated for this example.

The state assignment fault chart shown in Figure 4.8(b) contains only that information which was not obtained on the output assignment fault chart. Specifically, the

OUTPUT ASSIGNMENT FAULT CHART

FAULT		INPUTS (xy_1y_2)								REMARKS
ELEMENT	FAULT	R.S.	000	101	111	010	101	011	100	GOOD
		0	0	1	1	0	1	1	0	
GATE 1	s-a-0	0	0	0						2-0
GATE 2	s-a-0	0	0	1	1	0	1	0		6-0
GATE 3	s-a-0	0	0	0						2-0
GATE 1	s-a-1	1								1-1
GATE 2	s-a-1	1								1-1
GATE 3	s-a-1	1								1-1
TIME		0	1	2	3	4	5	6	7	

R.S. = r = reset

Figure 4.8(a)

STATE ASSIGNMENT FAULT CHART

ELEMENT	FAULT	R.S.	000	101	111	010	101	011	100	GOOD
		0	0	1	1	0	1	1	0	
GATE 4	s-a-0	0	0	0						2-0
GATE 4	s-a-1	0	0	1	1	1				4-1
TIME		0	1	2	3	4	5	6	7	

R.S. = r = reset

Figure 4.8(b)

NOT gate (gate number 4 in Figure 4.7(a)) is the only one included in the chart. A s-a-0 fault on this gate is discovered during the trigger pulse at time 2. The same indication is given by gate 1 and gate 3 in the s-a-0 condition.

The memory faults which are detectable through output observations are shown in Figure 4.8. There are six possible faults corresponding to the types shown in the table in Figure 4.5. Note that there is one fault (M-1 with trigger fault 1) which cannot be detected by the state exercising sequence selected. So although the sequence is a state exercising sequence, its length is insufficient to detect this fault.

It is apparent that certain of the faults will not always be detected or isolated by the state exercising sequence. For example, the gate s-a-1 faults for the output assignment all give the same indication for the R.S. pulse and for all succeeding inputs. It is possible to use a fault tree similar to that described in Chapter 1 (and 3) to determine the source of a given failure. In general, it may be observed that if two fault indications occur in the same time interval, it will be impossible to distinguish between the two faults. We are here assuming a single output terminal network. An alternative which we have mentioned is to extend the sequence to the end and see if any additional bits differ.

For the given example, all but one of the faults was detected by the given sequence. However, it is not possible to obtain isolation of a single element. Those faults giving the same incorrect output at the time interval are summarized below:

MEMORY FAULT CHART

ELEMENT	FAULT	R.S.	000	101	111	010	101	011	100	GOOD
		0	0	1	1	0	1	1	0	
M-1	s-a-0	0	0	1	1	0	1	0		6-0
M-2	s-a-0	0	0	0						2-0
M-1	s-a-1	0	0	1	1	1				4-1
M-2	s-a-1	0	0	1	1	1				4-1
M-1	T.F.1	0	0	1	1	0	1	1	0	
M-2	T.F.1	0	0	1	1	1				4-1
TIME		0	1	2	3	4	5	6	7	

Figure 4.9

SUMMARY

TIME	FAULT
1	Gate 1 s-a-1
	Gate 2 s-a-1
	Gate 3 s-a-1
2	Gate 1 s-a-0
	Gate 3 s-a-0
	Gate 4 s-a-0
	M - 2 s-a-0
4	Gate 4 s-a-1
	M - 1 s-a-1
	M - 2 s-a-1
	M - 2 T.F.1
6	Gate 2 s-a-0
	M - 1 s-a-0

This summary shows the time at which the output is different from the good output and lists the possible faulty elements. As mentioned, a test diagram could be developed to obtain fault isolation information by using the output for all time intervals.

Using the figure of merit given in the Introduction we have

$$\mu = \frac{TF.FD}{SL(TF - FD + 1)} \qquad TF = 14$$

$$FO = 13$$

$$= 13 \qquad SL = 7 \qquad 4.2$$

the maximum value of μ for any network will be TF.FD; μ can be normalized to 1 by dividing by TF.FD.

4.7 EMPIRICAL-EXPERIMENTAL METHODS FOR FAULT DETECTION

4.7.0 Introductory Comments

In many cases, fault detection equipment must be designed to work for networks already in operation. This means that the sequences used to detect faults must be developed after the network design has been completed. In some cases, the state table and/or network equations will not be available. For diagnosing faults in a network on which little of the detailed operating descriptions are known usually requires empirical-experimental methods.

Lee has developed a method for fault detection and isolation in digital sequential networks based on empirical methods.[79] Detecting the absence of a correct signal in a sequence or the presence of an incorrect signal in the sequence is the basis of his technique. An ensemble of signals which appear in the system is selected which he calls key events. The order of occurrence of these signals is important. If a network fault occurs, it is assumed that a departure from the prescribed sequence will follow. The incorrect sequence can be detected by feeding the ensemble into a gating circuit which compares the set of gated signals with a copy of the good gates signals. A fault can be indicated by an EXCLUSIVE OR of the copy with the actual network signals. By noting the time at which a fault is indicated it is possible to determine the incorrect line on the gating which will in turn be related to some specific fault within the network. Lee states that test diagrams can be used to develop fault isolation methods from the given data.

The basic drawback of Lee's method is that it requires special hardware for its implementation. In addition, its effectiveness cannot be estimated a priori. Manning has described an empirical-experimental method for self-diagnosis in a computing system[91]. He applies both simulation and detailed analysis of the properties of the computing system to arrive at a sequence of tests for self-diagnosis of the particular machine. He makes many conclusions. Among these is the fact that an asynchronous, level logic, cascade-organized machine is the most amenable to self-diagnosis. His method depends on a simulation program devised by Seshu[126]. Jones and Mays[65] have described a technique for verifying the integrity of large scale integrated sequential network blocks. The method which they used is what they describe as an "integrated approach". They employ the best features of Seshu's technique and add a few of their own. A six gate sequential network which they use as an example required 19 input combinations to detect all possible one-at-a-time faults. They have assumed stuck-at faults only.

4.7.1 Adapting Pseudo Correlation to Sequential Network Testing

In cases where the analysis method described in Section 4.6 cannot be applied because the networks are too large, the state tables are not available or time does not permit the detailed analysis, the method described in this section can be used.

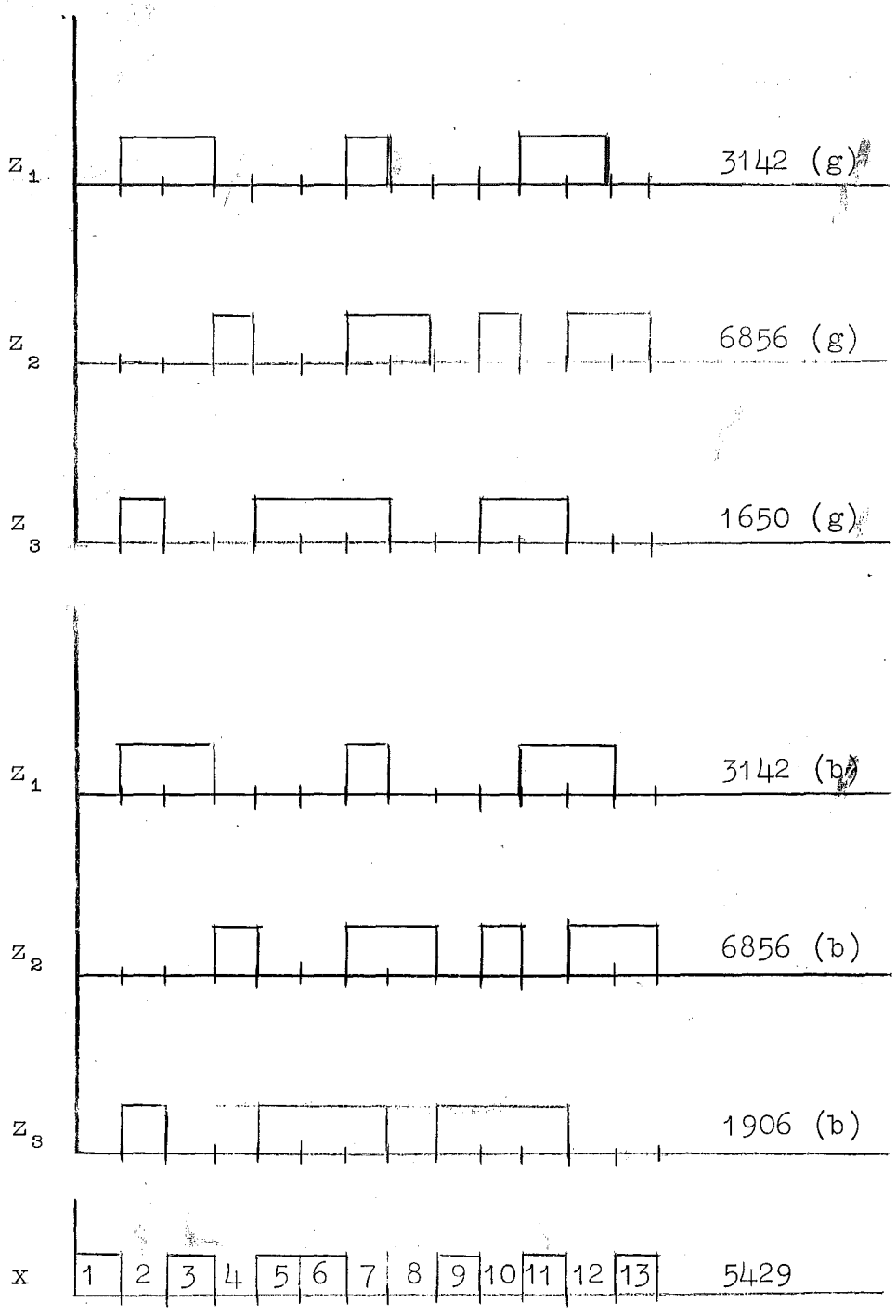
The method presupposes the availability of a good copy of the network in its physical realization or a model of the network. The model is preferred because it

can be used to develop a computer program which contains a fault simulation routine. This permits possible input sequences to be evaluated for their ability to detect faults of various types. The method is essentially identical to the pseudo correlation method described in Chapter 2 for analogue networks. It is assumed that a sequence of known input symbols is applied to the sequential network and that one of the outputs is monitored. If the network is good and the initial state can be set, the output sequence obtained for the given input sequence will be of a particular form. Again, we assumed that a fault of some type will cause a deviation in the output signal for the same input sequence. This is Lee's assumption. If no deviation is observed, the network is judged good. If a deviation occurs, the sequence will have at least one output symbol (bit) different from the good sequence. If sufficient input symbols are applied or if more than one output terminal is available, then a fault will cause at least one of the output bits to be different from the good network response.

There are various ways of measuring the output sequence but the one which we suggest is based on the expression

$$\psi = \sum_{n=o}^{M} 2^n Z(n\Delta t) \qquad 4.3$$

where the weighting function w (see Chapter 2) is 2^n, M is the output sequence length and Z is the output. Because the value of ψ is a real number, it can easily be incorporated into a fault dictionary for a posteriori diagnosis. As an example of this method, Figure 4.10 shows the output signals for a good network and for a network with a fault. The fault is indicated on output z_3



KEY: g good
b fault

Psuedo correlation giving fault indication output z_3 during time interval 9.

Figure 4.10

and occurs during the output time t = 9. This gives a pseudo correlation value of the signal of 1906 rather than the value of 1650 for the network without the fault.

Information of the type shown in Figure 4.10 can be developed by manual fault insertion[90] but preferably by a computer simulation using network element models. It is possible to use the method for very large networks by breaking the network at suitable terminals and simulating the faulty network. An input sequence which gives an output which differs from the output for the good network can be developed by an exhaustive method or by interaction with the computer program. The outputs from the faulty network can then be applied to the successive networks and the observable output can be pseudo correlated. If a discrepancy is noted, the fault is detectable. If the pseudo correlation gives a value which is different from all other values for all other faults, the fault can be isolated. The idea can be illustrated by the network shown in Figure 4.11. There are four sub-networks shown connected by the lines (there may be several wires per line) a,b, and c. The outputs a and b can be used as inputs to the network 3 and the output sequence c and b used as the inputs to network 4. If the network number 3

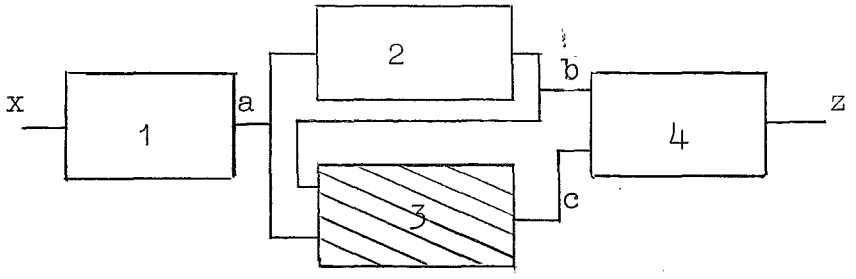


Figure 4.11

is faulty, the various faults will alter the output sequence on line c. Using a fixed sequence on x, the various faults in 3 can be simulated and the output c can be fed into network 4 along with the sequence on b. For fault detection, a sequence x which gives an output sequence z which is incorrect at one point in time will be sufficient. If one of the symbols in c is incorrect for a fault in sub-network 3, this faulty sequence can be used along with the good sequence on b to stimulate sub-network 4. If the fault is output observable, at least one bit of the output sequence on z will be incorrect.

It is possible then to use a simulation to develop faulty output sequences for the various network elements and test the individual elements for their processing of the faulty sequences. In many cases, the insertion of a faulty bit at any point in the sequence will cause the output to be in error. If it does not, then the fault is not output detectable by the given input sequence x. It could be argued that if the network is non-redundant and if there are no equivalent states, then the output must be different for some input sequence if there is a fault in one of the sub-networks.

4.8 SUMMARY AND CONCLUSIONS

This chapter has discussed the diagnosis of synchronous sequential networks. The state exercising sequence and state variable observability have been defined and their application to diagnosis explained.

The advantage of considering the actual network components when developing diagnostic information is the possibility of finding shorter test input sequences.

Because the checking experiments developed by Hennie depend on verifying all state transitions, they cannot be efficient. The state exercising sequence is potentially efficient because the network circuitry is considered.

If a sequential network is to be designed for efficient diagnosis, the states should be observable in the sense which we have defined. In addition, a synchronous design which has a state assignment which causes all memory units to change state during one time interval is potentially useful for diagnosis of memory transition faults.

Although the technique for detecting faults which was described in Section 4.6.2 is recommended, the experimental-empirical technique can be used for a priori development when the more analytical method cannot be applied.

The ideas in this chapter are still in primitive form. It is believed that the state exercising sequence is worthy of further investigation. State variable observability should be "designed in" when possible.

Although we have not considered the possibility here, there are strong arguments for constructing canonical MSP or MPS[77] output and state assignment networks to realize optimal sequential network diagnostic tests. This is feasible with cheaper LSI networks now being developed. Hence, the idea of <u>minimal</u> networks may be only one consideration in the design process.

REFERENCES & BIBLIOGRAPHY

1. ABEND K., HARLEY T.J. and KANAL L.N. — Classification of Binary Random Patterns. IEEE Trans. on Information Theory Vol. IT-11. No.4. October 1965, pp.538-543.

2. ARBIB M. — "Brains, Machines and Mathematics". McGraw Hill, 1964.

3. ARBIB M.A. and ZEIGER H.P. — An Automata Theoretic Approach to Linear Systems. IFAC Symposium on System Dynamics and Automatic Control in Basic Industries - Sydney 1968.

4. AKERS S.B. — On a Theory of Boolean Functions, Journal of Soc. Ind. Math. Vol. 7. No.4. Becemon 1959.

5. AMAR V. and CONDULMARI N. — Diagnosis of Large Combinational Networks. IEEE (TEC) Oct. 1967 pp.672-679.

6. AMARI S. — Theory of Information Spaces Transformations of Metric Signal Spaces. English Trans. Electronics Communication in Japan, Vol. 48 No.10. Oct. 1965 pp.96-107.

7. ARMSTRONG D.B. — On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Networks, IEEE Trans. on El. Comp., Vol. EC-15, No.1. Feb. 1966.

8. ARNOLD B.H. — "Intuitive Concepts in Elementary Topology". Prentic Hall 1962.

9. ASHBY W. ROSS — Measuring the Internal Informational Exchange of a System. CYBERNETICA (Namur) Vol. VIII No.1. 1965.

10. BANTA, EDWIN D. — On the Autocorrelation Function of Quantized Signal Plus Noise. IEEE Trans. On Information Theory. Jan. 1965 pp 114-117.

11. BASHKOW T.R. — A Programming System for Detection of Machine Malfunctions. IEEETEC Feb. 1963.

12. BAZOVSKY I. — "Reliability Theory and Practice". Prentice Hall Technology Series 1961.

13. BERKOWITZ R.S. — Conditions for Network-Element-Value Solvability. IRE Trans. on Circuit Theory. March 1962 pp.24-29.

14. BOOTH T.L. — An Analytical Representation of Signals in Sequential Networks. Proceedings of the Symposium on Mathematical Theory of Automata. Vol. XII April 24-26, 1962. Polytechnic Press, Brooklyn, N.Y. 1963 pp.301-340.

15. BOURICISU W.G., CARTER W.C., ROTH J.P. and SCHNEIDER R.R. — On-Line Reliability Calculations to Achieve a Balanced Design of an Automatically Repaired Computer, IBM Research R.C. 1800 Ap.11, 1967.

16. BRAZILOVICH Y.Y. — Determination of Optimum Periods of Preventive Maintenance in Automatic Systems. Translation from Russian Eng. Cybernetics. No.3. May-June 1964.

17. BROOME P.W. and YOUNG F.J. — The Selection of Circuit Components for Optimum Circuit Reproducibility. IRE Trans. on Circuit Theory. March 1962 pp.18-23.

18. BRUER M.A. — Design Automation of Digital Computers. Proc. IEEE Dec. 1966.

19. BRULE J.D., JOHNSON R.A. and KLETSKY E.J. — Diagnosis of Equipment Failures. IRE Trans. on Reliability and Quality Control. Vol. RQC-9, pp.23-24, April 1960.

20. BRZOZOWSKI A. — A Survey of Regular Expressions and Their Applications. IRE Trans. on Electronic Computers. June 1962. pp. 324-335.

21. BROZOZOWSKI J.A. — Regular Expressions from Sequential Circuits. IEEE (TEC) Vol. EC-13 Dec. 1964. No.6. pp.741-744.

22. CIANIELLO E.R. "Automata Theory". Academic Press, New York 1966. A Book on the Conference in 1964.

23. CHANG H.Y. An Algorithm for Selecting and Optimum Set of Diagnostic Tests. IEEE (TEC) Vol. EC-14. No.5. Oct. 1965 pp.706-711.

24. CHERNIKOVS.A. Investigation of Non-Linear Automatic Control Systems with External Distrubance pp.130-143. Eng.-Cybernetics, No.3. May-June, 1964.

25. CHESTNUT H. "Systems Engineering Tools". John Wiley and Sons Inc. 1965.

26. CHU W.W. A Mathematical Model for Diagnosing System Failures. IEEE Trans. on Elec. Comp. Vol. EC-16 No.3. June, 1967.

27. COOPER P.W. The Hyperplane in Pattern Recognition. CYBERNETICA (Namur) Vol.V. No.4. 1962.

28. COX P., and RANKIN K.F. Reliability in Large Electronic Data Processing Systems. A.T.E. Journal. Vol. 21. No.4.

29. DAVIES A.C. Signal Flow Graphs and Linear Feed-back Shift Registers Letter IEEE EC Jan. 1967.

30. DEO N. The Self-Diagnosability of a Computer. IEEE (TEC) Vol. EC. No.5. Oct. 1966 p.799.

31. DENT J. Diagnostic Engineering IEEE Spectrum. July 1967. pp.99-104.

32. DETTMAN, J.W. "Mathematical Methods in Physics and Engineering". McGraw Hill, 1962. N.Y.

33. ECKMAN D.P. (ED.) "Systems, Research and Design". Proceedings of the First Systems Symposium at Case Institute of Technology - Wiley 1961.

34. EICHELBERGER E.B. — Hazard Detection in Combinational and Sequential Switching Circuits, Proc. 5th Symposium on Switching Circuit Theory and Logical Design 1964 also in IBM J. Res. and Dev. Vol.9. 1965.

35. ELGERD O.I. — "Control Systems Theory". McGraw Hill, N.Y. 1967.

36. FEL'DBAUM A.A. — Application of the Theory of Statistical Solutions to Open - and Closed - loop Automatic Control Systems. No.1. Eng.-Cybernetics, Jan-Feb. 1963.

37. FRANK H. and YAU S.S. — Improving Reliability of a Sequential Machine by Error Correcting State Assignments. IEEE Trans. On Elect. Comp. (Short notes) pp.111-112 Feb. 1966.

38. FRIEDMAN A.D. — Fault Detection in Redundant Circuits. IEEE Trans. EC. Feb. 1967.

39. GALEY J.M. — Diagnosis of Failures in Sequential Circuits. Research Memorandum SR-157 IBM Corp. April 24 1961.

40. GALEY J.M., NORBY R.E. and ROTH J.P. — Techniques for the Diagnosis of Switching Circuit Failures. IEEE Trans. on Comm. and Elec. pp.509-514. No.70. Sept. 1964.

41. GERSHKOVICH Y.B. and POLTEROVICH V.M. — Nonrepeating Superpositions of Boolean Functions, Automatika i Telemekhanika. No.5. pp.109-152 May 1967.

42. GERTSBAKH I.B. — Optimum Rule for Maintenance of a System with Many States. Engineering Cybernetics. No.5. Sept.-Oct. 1966. pp.75-80. Translated from Teknicheskaja Kibernetika.

43. GILL A. — "Introduction to the Theory of Finite-State Machines". McGraw-Hill, New York 1962.

44. GILL A. and FLEXER J.R. — Periodic Decomposition of Sequential Machines. J. at the Ass. of Computing. Vol.14. No.4. 1967.

45. GLUSS B. — An Optimum Policy for Detecting a Fault in a Computer System. The Journal of the Op. Aut. Soc. of Am. Vol.7. No.4. July-Aug. 1959.

46. GRIESMER J.H. — A Method for the Diagnosis of Failures in a Combinational Circuit, IBM Research SR-158, April 25 1961.

47. GOSLING W. — "The Design of Engineering Systems". Heywood and Co. Ltd. 1962, London.

48. GOLDMAN A.S. and SLATTERY T.B. — "Maintainability - A Major Element of System Effectiveness". Wiley 1964.

49. HADLOCK F. — On Finding a Minimal Set of Diagnostic Tests. IEEE Trans. on Elec. Comp. pp.674-679. Oct.1967.

50. HALL A.D. — "A Methodology for Systems Engineering". D Van Nostrand Co. Ltd. 1962.

51. HAPP W.W. — Combinatorial Analysis of Multi-Terminal Devices. IEEE SC June 1967.

52. HARDIE F.H. and SUHOCKI R.J. — Design and Use of Fault Simulation for Saturn Computer Design. IEEE TEC Vol. E.C. 16. No.4. August 1967.

53. HARTMANIS J. — Loop-free Structures of Sequential Machines. Information and Control, Vol.5. No.1. March 1962. pp.25-43.

54. HARTMANIS J. — On Memory Requirements for Context Free Language Recognition. Journal of the Association for Computing Machinery. Vol.14. No.4. 1967.

55. HASHIMOTO A., KASAMI T., OZAKI A. — Fault Diagnosis in Combinational Logical Networks, Elec. and Comm. in Japan. Vol.47. No.4. April 1964.

56. HENNIE F.C. — "Finite State Models for Logical Machines". John Wiley and Sons 1968.

57. HOBERMAN S. "Trouble Shooters Reference Handbook". Foulsham and Co. 1963.

58. HOLICK A. Analysis of Noncatastrophic Failures in Digital Guidance Systems. IEEE Aug. 1963 TEC. pp.365-371.

59. HOPCROFT J.E. and ULLMAN J.D. An Approach to a Unified Theory of Automata B.S.T.J. Oct. 1967. Vol.XLVI. No.8.

60. HUFFMAN D.A. The Synthesis of Sequential Switching Circuits. Journal of the Franklin Institute. Vol.251. pp.161-190, 275-303.

61. HUFFMAN D.A. Canonical forms for information-loss less finite-state logical machines. (Transactions of the I.R.E. Professional Group on Circuit Theory. Vol.CT-6 Special Supplement. May 1959. pp.41-59.

62. INCE E.L. Ordinary Differential Equations. pp.408-415. Longmans Green and Co. Inc. New York 1927.

63. IRESON G. "Reliability Handbook". McGraw-Hill, 1966.

64. JELINEK H. and BREEN H. Proper Test Point Allocation. Electro-Technology, June 1966.

65. JONES E.R. and MAYS C.H. Automatic Test Generation Methods for Large Scale Integrated Logic. IEEE Journal of Solid-State Circuits. Vol.SC-2. No.4. December 1967.

66. JOYCE B.T. and STOCKTON E.M. "Computer Controlled Automatic Testing". Electro Technology. October 1966.

67. KALMAN R.E. Optimal Synthesis of Linear Sampling Control Systems using Generalized Performance Indices. pp.1820 - Trans. A.S.M.E. Nov. 1958.

68. KALMAN R.E. and BUCY R.S. New Results in Linear Filtering and Prediction Theory. Trans. A.S.M.E. March 1961. pp.95-108.

69. KAUTZ W.H. — Fault Testing and Diagnosis in Combinational Digital Circuits. IEEE (TEC) Vol.C-17. No.4. April 1968. pp.352-366.

70. KILMER W. — Iterative switching networks composed of Combinational Cells. IRE TEC April 1962. No.2. Vol.EC-11. pp.123-131.

71. KIME C. — An Organization for Checking Experiments on Sequential Circuits. IEEE Trans. on Elec. Comput. pp.113-115 (Short Notes). Feb. 1966.

72. KOHAVI Z. and LAVELLEE P. — Design of Sequential Machines with Fault Detection Capabilities. IEEE Trans. on Elec. Comp. Vol.EC-16. No.4. Aug. 1967.

73. KOHAVI I. and KOHAVI Z. — Variable Length Distinguishing Sequences and their Application to the Design of Fault Detection Experiments. IEEE Transactions on Computors Vol.C-17. No.8. Aug.1968.

74. KOLMOGOROV A.N. and FOMIN S.V. — Functional Analysis. Vol.2. Graylock Press Albany, New York 1961. Translated from Russian.

75. KRASOVSKIY A.A. — Variation in Entropy of Continuous Dynamic Systems. Eng. Cybernetics No.5. Sept.-Oct. 1963.

76. KUH E.S. and ROHRER R.A. — The State-Variable Approach to Network Analysis. Proceedings of the IEEE July 1965. pp.672-686.

77. KRIEGER M. — "Basic Switching Circuit Theory". Macmillan 1967.

78. KRUSKAL J.B. and HART R.E. — A Geometric Interpretation of Diagnostic Data from a Digital Machine. Based on a study of the Morris Illinois Electronic Central Office. The Roll system Technical Journal. Oct. 1966. pp.1299-1338.

79. LEE F. — An Automatic Self-Checking and Fault Locating Method. IRE Trans. on Elec. Comp. pp.265-267. April 1962.

80. LERNER S.B. Hazard Correction in Asynchronous Sequential Circuits. IEEE Trans. on Elec. Comput. pp.265-267. April 1965.

81. LEVADI V.S. Automated Learning Applied to Fault Diagnosis. IEEE Trans. on Aerospace and Elec. Systems. Vol.AES-3. No.6. Nov. 1967.

82. LIPSCHUTZ S. "Set Theory and Related Topics". Schaum Publishing Co. New York 1964.

83. LOFGREN L. Kinematic and Tessellation Models of Self-Repair. Techn. Report No.8. Engineering Expt. State University of Illinois. Dec. 1961.

84. LOFGREN L. Self-Repair as a Computability Concept in the Theory of Automata. Proceedings of the Symposium on Mathematical Theory of Automata. Vol.XII. April 24-26 1962. Polytechnic Press Brooklyn N.Y. 1963. pp.205-223.

85. LYUBATOV YU.V. Optimal Procedures for Localization of Breakdown In a Modularized Radioelectronic System. (Russian Translation) Engineering Cybernetrics No.4. July-Aug. 1964. pp.14-21.

86. MCCLUSKEY E.J. Minimization of Boolean Functions. Bell Sys. Tech. J. Vol.XXXV. No.6. Nov. 1956.

87. MALING K. and ALLEN E.L. A Computer Organization and Programming System for Automated Maintenance. IEEE Trans. on E.C. Vol.EC-12. pp.887-895. Dec. 1963.

88. MANDELBAUM D. A Measure of Efficiency of Diagnostic Tests upon Sequential Logic. IEEE (TEC) EC-13. Oct. 1964. No.5. p.630.

89. MANNING E. and CHANG H.Y. A Comparison of Fault Simulation Methods for Digital Systems. Digest 1st Annual Computer Conference Chicago. Ill. Sept. 6-8 1967.

90. MANNING E. and CHANG H.Y. — Functional Techniques for Efficient Digital Fault Simulation. Digest IEEE Int. Convention. Paper 7B-2. 1968.

991. MANNING E. — On Computer Self-Diagnosis Parts I and II. IEEE TEC. Vol.EC-15. No.6. Dec. 1966.

92. MEALY G.H. — A Method for Synthesizing Sequential Circuits. Ball System Tech. Journal 34. pp.1045-1079. Sept. 1955.

93. MEISEL W.S. and COLLINGS D.C. — Hazards in Non-Critical Races. IEEE Proceedings Letters. Aug. 1968. pp.1361-1363.

994. MESAROVIC M.D. — "Views on General Systems Theory". John Wiley and Sons. 1964.

95. MESAROVIC M.D. — "The Control of Multivariable Systems". Wiley, N.Y. 1960.

96. MIKUSINSKI J. — "Operational Calculus". Pergamon Press. 1959.

97. MISCHKE C.R. — "An Introduction to Computer Aided Design". Prentice-Hall. 1968.

98. MITCHELL B.A. — A Hybrid Analog-Digital Parameter Optimizer for Astrac II Proceedings-Spring Joint Computer Cr. 1964. pp.271-285.

99. MINE H. and KOGA Y. — Basic Properties and a Construction Method for Fail-Safe Logical Systems. IEEE Trans. on Elec. Comp. Vol.EC-16. No.3. June 1967.

100. MINSKY M. — "Computation Finite and Infinite Machines". Prentice-Hall. 1967.

101. MINTZ M.L. and THORP J.S. — A Method for Discrete System Identification and Approximation. Journal Franklin Inst. No.5. May 1967.

102. MIRKIN B.G. — Identification of the Relative Equivalence of States in Sequential Machines. Automation and Remote Control (Russian Translation). 1967. No.2.

103. MISHKIN E. and BRAUN L. "Adaptive Control Systems". McGraw-Hill, N.Y. 1961.

104. MITCHELL B.A. A Hybrid Analog-Digital Parameter Optimizer for Astrac II Proceedings-Spring Joint Computer Conf. 1964. pp.271-285.

105. MOE M.L. and MURPHY G.J. An Approach to Self-Adaptive Control Based on the Use of Time Moments and a Model Reference. IRE Trans. on Automatic Control. Oct. 1962. pp.82-92.

106. MOORE E.F. (EDITOR) "Sequential Machines". Selected Papers. Addison-Wesley. Reading, Mass. 1964.

107. MOORE E.F. "Gedanken - Experiments on Sequential Machines", in Automata Studies. pp.129-153. Princeton University Press, Princeton, N.J. 1956.

108. NAJJAR H.F. Partition Techniques for Switching Logic. Electro-Technology. April 1967. pp.50-54.

109. NEKULA N.N. Automatic Minimization of Boolean Functions, Automatika i Telemekhanika. No.5. pp.80-85. May 1967.

110. PATTERSON G.W. and MCNAUGHTON R. Combinational Elements and Fiducial-State Assignments. Proceedings of the Symposium on Mathematic Theory of Automata. Vol.XII. Apr.24-26 1962. Polytechnic Press, Brooklyn, N.Y. 1963. pp.459-481.

111. PEGIS R.J., GREY D.S., VOGL T.P and RIGLER A.K. The Generalized Orthonormal Optimization Program and its Application, from Recent Advances in Optimization Techniques by Lavie Vogl. Wiley. 1966. pp.47-60.

112. PATERSON E.L. "Statistical Analysis and Optimization of Systems". Wiley. 1961.

113. PETERSON D.W. and MATTSON R.L. — A Method of Finding Linear Discriminant Functions for a Class of Performance Criteria. Linear Discriminant Functions. 1966. pp.380-387.

114. PIPES L.A. — "Applied Mathematics for Engineers and Physicists". McGraw-Hill, 2nd Ed. 1958.

115. POAGE J.F. — Derivation of Optimum Tests to Detect Faults in Combinational Circuits. Proceedings of the Symposium on Mathematical Theory of Automata. Vol.XII. April 24-26 1962. Polytechnic Press, Brooklyn, N.Y. 1963. pp.483-528.

116. POLOUKO A.M. — "Fundamentals of Reliability Theory". Academic Press 1968.

117. PURI N.N. and WEYGANDT C.N. — Transfer Function Tracking of a Linear Time Varying System by Means of Auxiliary Simple Lag Networks. Joint Automatic Cont. Conf., Paper VIII-2. 1963.

118. ROE P.O'N. — "Networks and Systems". Addison-Wesley. 1966.

119. ROTH J.P. et.al. — Investigations in the Design of an Automatically Repaired Computer, Private Communication (to be published).

120. ROTH J.P. — Diagnosis of Automata, IBM Research SR-114. July 21 1960.

121. ROTH J.P. — On Computing Diagnostic Tests for Circuits with Feedback, IBM Research SR-140. Nov.29 1960.

122. ROTH J.P. — Diagnosis of Automata Failures: A Calculus and a Method. IBM Journal of Research and Development. Vol.10. No.4. July 1966.

123. ROTH J.P. — Diagnosis of Automata Failures. Research Memorandum SR-114 IBM Corp. Poughkeepsie, N.Y. July 1960.

124. SANTOS J. and ARRANGO H. — On the Analysis and Synthesis of Three-Valued Digital Systems. Proceedings-Spring Joint Computer Conference 1964. pp.463-475.

125. SELLERS F.F., HSIAO M.Y. and BEARNSON L.W. — Error Detecting Logic for Digital Computers. McGraw-Hill 1968.

126. SESHU S. — On an Improved Diagnosis Program. IEEE Trans. on Elec. Comp., IC-14. No.1. Feb. 1965.

127. SESHU S. and WAXMAN R. — Fault Isolation in Conventional Linear Systems. IEEE Trans. on Reliability. Vol.R-12. No.1. May 1966. p.11/16.

128. SESHU S. and FREEMAN D.N. — The Diagnosis of Asynchronous Sequential Switching Systems. IRE Trans. on Electronic Computers. Vol.EC-11. pp.459-465. Aug. 1962.

129. SHAPIRO G. ROGERS G.J. LAUG O.B. and FULCOMER P.M. — Fault Isolation by Semiautomatic Techniques. Part I - Basic Concept and Techniques. IEEE Spectrum. Aug. 1964. pp.98-113.

130. SHINNERS S.M. — "Techniques of System Engineering". McGraw-Hill 1967.

131. SPRAGINS J. — A Note on the Iterative Application of Bayes' Rule. IEEE Trans. on Information Theory. Oct. 1965. pp. 544-549. Vol.IT-11. No.4.

132. TOU J. — Digital and Sampled-Data Control Systems. McGraw-Hill 1959.

133. TAUSSKY OLGA — On the Variation of the Characteristic Roots of a Finite Matrix Under Various Changes of its Elements. On the Variation of the Characteristic Roots. pp.125-138.

134. TERNO O.R. — Hybrid Functions - A New Method for Characterization of Complex Systems. May 1965. pp.12-17.

135. TIMONEN L.S. — Construction of Optimal Programs for Diagnostics, Russian Trans. IEEE Around Mid 1966. Eng. Cybernetres No.4 July-Aug. 1966 pp.94-100. Translated from Teknicheskay a Kibernetika.

136. TRUITT T.D. — Hybrid Computation... What is it? ... Who Needs it?... . Proc. Spring Joint Computer Conference, 1964 pp. 249-269.

137. TSIANG S.H. and ULRICH W. — Automatic Trouble Diagnosis of Complex Logic Circuits. B.S.T.J. Vol. XLI. July 1962 No.4.

138. UDAGAWA K., INAGIKI Y. and TANGE H. — The State Characteristic Equations of Finite Automata and Their Regular Expressions. Electronics and Communication in Japan, Vol.48, No.9, Sept. 1965.

139. UDAGAWA K. and INAGIKI Y. — Probablistic Studies of Logical Circuits and the Synthesis of Reliable Circuits Using Probablistic Logical Matrix. Electronics and Communication in Japan. Vol. 46, No.11. Nov. 1963.

140. VALSTAR J.E. — Fault Isolation of Electronic Circuits using Transfer Function Methods, U.S. Air Force Document, APL-TDR-64-123. Oct. 1964.

141. VALSTAR J.E. — In-Flight Dynamic Checkout. IEEE Trans. on Aerospace-Support Conference Procedures. Vol. AS-1 No.2. Aug. 1963. pp.213-221.

142. WEBB H. and BREEN H.T. — "Sycate, A Digital Computor Program". Private Communication.

143. WEITZENFELD A.S. and HAPP W.W. — Combinatorial Techniques for Fault Identification in Multiterminal Networks. IEEE T-Rel. Vol. R-16. No.3. Dec. 1967.

144. WEYGANDT C.N. and PURI N.N. — Transfer Function Tracking and Adaptive Control Systems., IRE Trans. on Auto Control. Vol. AC-6, No.2. May 1961.

EPILOGUE

The moving finger writes; and, having writ,
Moves on:
Nor all your piety nor wit, shall lure it back
To cancel half a line,
Nor all your tears wash out a word of it.

OMAR KHAYYAM

In this thesis, a step has been made along the path to formulating an approach to hybrid electrical system diagnosis. Many questions on the engineering significance of this work remain to be answered.

The formulation and solution of models which represent important facets of a particular physical phenomenon is the engineers forte. Because faults can alter the physical characteristics of a system, the formulation of models which apply to many variations of a network is usually more difficult than a single solution. Algebraic models based on continuous functions are potentially the most useful because they represent the totality of solution. The diagnostic model represents a large number of variations of a particular structure — component network.

Solution to many unsolved problems would contribute greatly to general system diagnosis which will be required for advanced systems. Some of the problems are as follows:

1. Specification of methods for partitioning the system for diagnostic analysis.
2. An investigation into the best organization for implementing diagnostic techniques. This would include consideration of testing equipment, test measurement methods and the placement of test points.
3. General precepts for "system design for diagnosis".

4. The development of interactive digital computer oriented diagnostic design methods. These would be similar to the computer aided performance design programs which are becoming increasingly more important.

These are but a few of the problems that require solving. Improvements to the techniques presented here would be the next step in the continuation of this investigation. In addition, relationships between power system fault protection and hybrid electrical system diagnosis could well be looked at. For purposes of applying the approach proposed, electro-mechanical systems containing servo's, gyroscopes and perhaps some hydraulic or fluidic equipment could form another branch of investigation.